VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Aplikace pro hlasové ovládání bezpilotních dronů s využitím mobilního zařízení

Application for Drone Voice Control Using a Mobile Device

2017 Bc. Miroslav Beleš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Miroslav Beleš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Aplikace pro hlasové ovládání bezpilotních dronů s využitím mobilního zařízení
Application for Drone Voice Control Using a Mobile Device

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je navrhnout, implementovat a následně otestovat aplikaci pro mobilní zařízení, která bude umožňovat zadávat hlasové příkazy pro bezpilotní drony, za účelem ovládání pohybu drona, či jeho příslušenství.

Zadání:

1. Detailně nastudujte problematiku hlasového ovládání bezpilotních modelů.
2. Proveďte rešerši stávajících řešení pro hlasové ovládání periferií bezpilotních modelů s využitím mobilních zařízení.
3. Navrhněte komunikační schéma mezi mobilním zařízením a modelem včetně definice možných technologií a protokolů.
4. Vytvořte aplikaci pro hlasové ovládání periferií bezpilotního modelu na platformě OS Android.
5. Realizujte testování funkčnosti aplikace a proveďte optimalizaci.
6. Vytvořte progamovou a uživatelskou dokumentaci k vytvořené aplikaci.

Seznam doporučené odborné literatury:

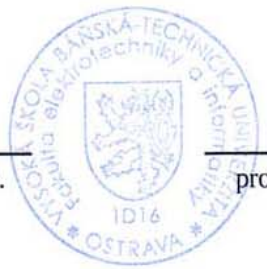
1. Zigurd Mednieks, Laird Domin, G. Blake Meike, Programming Android: Java Programming for the New Generation of Mobile Devices, ISBN-13: 978-1449316648.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017



doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 19. dubna 2017 .....

Rád bych touto cestou poděkoval především Ing. Filipu Řezáčovi, Ph.D. za odborné vedení práce, veškerý věnovaný čas, věcné připomínky a rady.

Abstrakt

Práce se zabývá problematikou hlasového ovládání příslušenství dronů, za pomoci mobilního zařízení s operačním systémem Android.

Teoretická část se věnuje teorií systémů pro automatické rozpoznávání řeči, včetně popisu architektury těchto systémů. V této části je popsáno využití skrytých Markovových řetězců a umělých neuronových sítí na poli systémů pro automatické rozpoznávání řeči.

Praktická část se zabývá implementací aplikace pro operační systém Android, která má za úkol rozpoznat hlasově zadávané příkazy pro ovládání kamery dronu a tyto příkazy převést na instrukce určené pro dron. Dále také popisem využívaných knihoven a vývojářských sad. Práce je zakončena testováním a optimalizací navrženého řešení.

Klíčová slova: Automatické rozpoznávání řeči, dron, UAV, Android, Parrot Bebop 2, umělé neuronové sítě, skryté Markovovy modely, CMU Sphinx

Abstract

This thesis is about speech controlled drone with use of mobile device that runs on Android operating system.

Theoretical part deals with theory of automatic speech recognition systems including its architecture. Hidden Markov models and artificial neural networks are described in this section as an approaches to systems for automatic speech recognition.

Converting speech commands to an instructions for drone control in Android operation system are described in practical implementation part. This section also includes testing and optimization.

Key Words: Automatic speech recognition, drone, UAV, Android, Parrot Bebop 2, artificial neural networks, hidden Markov models, CMU Sphinx

Obsah

Seznam použitých zkratek a symbolů **8**

Seznam obrázků **10**

Seznam tabulek **11**

Úvod **12**

1 Historický vývoj dronů a jejich principy **14**
1.1 Počátek éry multikoptér . . . 15
1.2 Princip fungování multikoptér a jejich komponenty . . . 15
1.3 Komunikace vysílač - přijímač . . . 19

2 Lidská řeč a její automatické rozpoznávání **21**
2.1 Základní architektura ASR systému . . . 23
2.2 Využívání skrytých Markovových modelů . . . 26
2.3 Shrnutí ASR systémů . . . 28

3 Umělé neuronové sítě **30**
3.1 Neurony . . . 30
3.2 Struktura neuronových sítí . . . 31
3.3 Trénování umělých neuronových sítí . . . 32

4 Hlasové ovládání bezpilotních modelů **34**
4.1 Stávající řešení dronů ovládaných hlasem . . . 34
4.2 Dron Parrot Bebop 2 . . . 36
4.3 Rozpoznávání hlasu pomocí CMU Sphinx . . . 40

5 Implementace aplikace Andron **43**
5.1 Struktura aplikace . . . 44
5.2 Návrh grafického rozhraní . . . 45
5.3 Implementace rozpoznávače hlasu . . . 47
5.4 Přiřazení hlasových příkazů k funkcím . . . 48
5.5 Video stream z dronu . . . 50
5.6 Testování a optimalizace . . . 51

Závěr **55**

Literatura **56**

Přílohy **57**

A Uživatelská dokumentace **58**

A.1 Popis jednotlivých obrazovek aplikace 58

A.2 Hlasové příkazy aplikace 61

B Programová dokumentace **63**

Seznam použitých zkratek a symbolů

ANN	–	Artificial Neural Network
ASR	–	Automatic Speech Recognition
ESC	–	Electronic Speed Control
DSSS	–	Direct Sequence Spread Spectrum
FHSS	–	Frequency Hopping Spread Spectrum
HHC	–	Human-Human Communication
HMC	–	Human-Machine Communication
HMM	–	Hidden Markov Model
ISM	–	Industrial, Scientific, Medical
NOTAR	–	NO TAil Rotor
RC	–	Radio Controlled
UAV	–	Unmanned Aerial Vehicle

Seznam výpisů zdrojového kódu

4.1 Úryvek kódu pro sestavení textového řetězce ze sériového toku 36
4.2 Zjednodušená ukázka implementace metody takeOff() 40
4.3 Ukázka definování slov ve slovníku 42
5.1 Implementace metody setupRecognizer() 48
5.2 Úryvek těla metody onPartialResult() 49
5.3 Úryvek těla metody moveCamera() 50
5.4 Úryvek layoutu aktivity SkyControllerActivity 51

Seznam obrázků

1.1 Kettering Bug 14
1.2 Patent multikoptéry 15
1.3 Brushless motor 17
1.4 Směr otáčení vrtulí 18
2.1 Vznik akustické vlny 22
2.2 Systém mluveného jazyka 23
2.3 Architektura ASR systému 24
2.4 Gramatika jako orientovaný graf 26
2.5 Pětistavový skrytý Markovův model 27
2.6 Zřetězené modely fonémů 27
2.7 Záznam zvukové nahrávky 28
3.1 Biologický neuron 30
3.2 Formální neuron 31
3.3 Struktura neuronové sítě 32
3.4 Typy neuronových sítí podle způsobu učení 32
4.1 Architektura hlasově ovládaného dronu IRIS+ 34
5.1 Blokové schéma projektu 43
5.2 Vztahy mezi aktivitami 45
5.3 Hlavní obrazovka 46
5.4 Obrazovka nastavení 46
5.5 Obrazovka seznamu zařízení pro připojení 46
5.6 Obrazovka s video streamem z dronu 46

Seznam tabulek

1.1 Přehled technologií RC vysílačů 20
4.1 Přehled informací získaných z mDNS 37
4.2 Dostupné klíče pro spojení 38
4.3 Dostupné klíče pro odpověď na požadavek na spojení 38
4.4 Seznam klíčových slov a frází v souboru keywords.list 42
5.1 Zařízení, na kterých proběhlo testování aplikace 51
5.2 Výsledky testování 52
5.3 Počet úspěchů pro jednotlivé příkazy u daných tetovacích sestavení 53

Úvod

Inspekce mostů, větrných a slunečních elektráren, různá měření v místech člověku nedostupných či nebezpečných nebo třeba filmařství – to je jen několik příkladů, jak a kde se dají v dnešní době využívat v průmyslových oblastech UAV zařízení (dnes většinou pod souhrnným názvem *dron*). Drony ke své činnosti, například při inspekcích mostních konstrukcí, většinou využívají kamery zavěšené na Kardanově závěsu, neboli gimbalu, nebo jsou vestavěny přímo do těla dronů. Pro ovládání kamer jsou na vysílačích většinou speciální páky a tlačítka pro otáčení kamery, zachycení fotografie nebo spouštění a zastavení nahrávání. Ovládání dronu, společně s kamerou, bývá náročné a je proto vyžadována přítomnost dalšího pilota, který se stará o ovládání příslušenství.

Tato diplomová práce si bere za cíl usnadnit pilotovi práci a redukovat počet osob obsluhujících dron na jednu. Cílem je umožnit pilotovi hlasově ovládat kameru vestavěnou na dronu pomocí mobilní aplikace. Velkou výhodou využití mobilního zařízení je zpětná vazba z dronu v podobě video streamu z kamery. Tím je zajištěno, že má pilot pod kontrolou jak dron, tak i ovládání a stav kamery.

Kapitola č. 1 shrnuje historii UAV zařízení a princip fungování multikoptér, včetně technologií použitých pro komunikaci mezi vysílačem a přijímačem. Jsou zde popsány všechny důležité komponenty každé multikoptéry.

Nedílnou součástí této diplomové práce je nastudování rozpoznávání hlasu počítačem. Tento obor má v dnešní době velký význam a můžeme se s jeho aplikacemi setkat v dnešních mobilních zařízeních v podobně mobilních asistentů. Hojně se hlasově ovládané aplikace využívají v lékařství, například pro nevidomé osoby. Své místo mají tyto aplikace například v projektech chytrých domů, kdy se může hlasově ovládat například řízení termostatu, či zapnutí některých spotřebičů. V kapitole č. 2 se věnuji charakteristice lidské řeči a podrobně vysvětluji architekturu systémů pro automatické rozpoznávání řeči.

Jednou z možností řešení automatického rozpoznávání řeči je využívání neuronových sítí. Kvůli obsáhlosti tohoto oboru jim proto věnuji celou kapitolu č. 3.

Praktická část této diplomové práce začíná kapitolou č. 4, ve které představím dron, na kterém vývoj probíhal, dále potom použité knihovny a vývojářské sady, využívané v tomto projektu, včetně příkladů jejich použití. Tato kapitola se zabývá také projekty souvisejícími s tématem této diplomové práce. Předem lze uvést, že má diplomová práce je oproti stávajícím řešením hlasově ovládaných dronů unikátní především tím, že cílem je ovládání periferií dronů, nikoliv dronů samotných.

Samotná implementace aplikace pro Android zařízení je popsána v kapitole č. 5. Popisuje se zde návrh a struktura projektu, dále také grafický návrh rozhraní a v neposlední řadě im-

plementace rozpoznávače hlasu a přiřazení jednotlivých hlasových příkazů k instrukcím pro ovládání kamery dronu. Neméně důležitou částí této diplomové práce je testování a optimalizace implementované aplikace. Tento aspekt je shrnut v závěru kapitoly č. 5.

V příloze práce je obsažena uživatelská a programová dokumentace. Uživatelská dokumentace se věnuje návodu na obsluhu, včetně popisu funkcí aplikace. V programové dokumentaci je zahrnuta adresářová struktura projektu a vygenerovaná Javadoc dokumentace.

1 Historický vývoj dronů a jejich principy

Pod názvem dron si mnoho z nás v současné době představí hlavně takzvané multikoptéry – kvadkoptéry, hexakoptéry nebo dokonce i oktokoptéry. Tyto multikoptéry se liší počtem svých rotorů a jsou v dnešní době velmi populární. Ovšem pod pojmem dron si můžeme představit jakýkoliv bezpilotní letoun (někdy také UAV z anglického Unmanned Aerial Vehicle), ať už je to letadlo, helikoptéra či již zmíněná kvadkoptéra.

Drony obecně mají velmi široké uplatnění, ať už při amatérském létání nebo v profesionální sféře – natáčení filmových snímků, v armádním prostředí nebo pro vědecké účely. Například Státní ústav radiační ochrany, veřejná výzkumná instituce, využívá bezpilotních dronů za normálních situací k měření radioaktivního záření ve špatně dostupných místech a při havarijních situacích, například k monitorování jaderné elektrárny nebo obecně při měření, kde by byl přístup člověka nebezpečný.

První bezpilotní letouny se objevily již v první světové válce, kde pod vedením společnosti Dayton-Wright Airplane Company vznikl bezpilotní letoun s názvem „Kettering Bug“ (obrázek č. 1.1), který dokázal létat s bombou vážící $81\,\mathrm{kg}$. Dříve, než mohl být tento letoun použit v praxi, válka skončila [8].

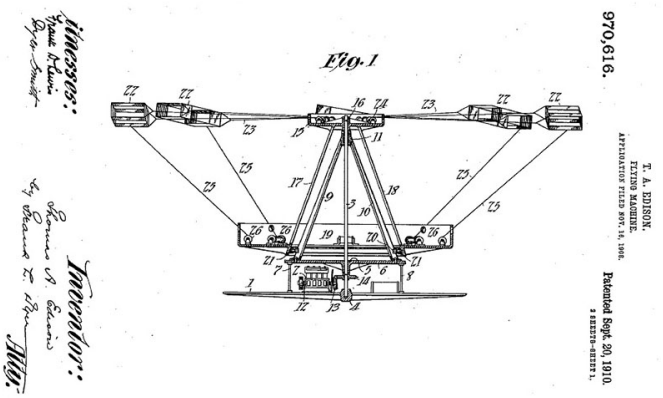
Obrázek 1.1: Kettering Bug (foto: Joe May)

S postupným technologickým pokrokem se staly drony více kompaktnějšími, lehčími, univerzálnějšími a především cenově dostupnějšími. S dobou se mění také používaná rádiová pásma pro komunikaci mezi vysílačem a přijímačem na dronu. V počátcích modelářství se využívala pásma $27\,\mathrm{MHz}$, $35\,\mathrm{MHz}$ a $40\,\mathrm{MHz}$. V současné době se v převážně využívá ISM pásma $2{,}4\,\mathrm{GHz}$. V minulosti bylo pro modelářství velkým problémem vzájemné rušení modelů, například na modelářských soutěžích. Každý modelář musel použít krystal, s odlišnou frekvencí než ostatní. Tato nevýhoda byla odstraněna díky technice rozprostření spektra DSSS a FHSS, kterou využívá pásmo $2{,}4\,\mathrm{GHz}$ (podrobněji kapitole č. 1.3).

1.1 Počátek éry multikoptér

Hlavním rozdílem mezi helikoptérou a multikoptérou je ten, že multikoptéra k manévrování nepotřebuje ocasní rotor ani trysky (u helikoptér s NOTAR systémem). Multikoptéru bychom tedy mohli definovat jako letadlo těžší než vzduch, které má dva nebo více, většinou symetricky umístěných rotorů a ovládá náklon podél osy x, y a z. Zdvih je řízen pomocí různých hodnot otáček rotorů, stabilizace je docílena kombinací elektromechanických senzorů a výpočetních jednotek.

První zmínka o multikoptéře pochází z hlavy vynálezce Thomase Alvy Edisona, který si multikoptéru nechal patentovat v roce 1908 pod názvem „Flying Machine“, tedy létající stroj. Patent, který byl uznán v roce 1910, je vyobrazen na obrázku č. 1.2.



Obrázek 1.2: Patent multikoptéry Thomase Alvy Edisona (zdroj: [9])

V současné době se pro pojem multikoptér vžil obecný název dron, který budu v této práci upřednostňovat. V případě nutnosti konkrétnějšího názvu využiji pojmu multikoptéra.

1.2 Princip fungování multikoptér a jejich komponenty

Princip fungování multikoptér a jehich komponenty pro stavbu, si budeme pro lepší přehled demonstrovat na kvadkoptéře, což je multikoptéra se čtyřmi rotory, kterou můžeme dále rozlišovat na kvadkopétru v konfiguraci X (kdy ve směru vpřed máme vepředu dva rotory a z pohledu shora připomíná písmeno X), plus (kdy ve směru vpřed máme vepředu jeden rotor a z pohledu shora připomíná symbol +) a H konfiguraci (kdy ve směru vpřed máme vepředu 2 rotory a pohled shora připomíná písmeno H otočené o 90°). Pro úspěšný vzlet kvadkoptéry potřebujeme hned několik klíčových komponent. V následujících odstavcích si je představíme.

Rám

Při stavbě kvadkoptéry je rám prvním rozhodovacím kritériem, které nám určí, jaké ostatní komponenty můžeme či nemůžeme používat. Rám nám určuje celkový vzhled

a velikost kvadkoptéry a zastává hlavní funkci kvadkoptéry, díky níž komponenty „drží pohromadě“. Při volbě rámu se musíme rozhodnout, jakou konfiguraci chceme pro stavbu použít (X, + nebo H). Rozměry rámu jsou měřeny u kvadkoptéry diagonálně, tedy od rotoru č. 1 k rotoru č. 3 (číslování uvažujeme podle obr. 1.4). V současné době jsou pro amatérské využití velmi populární rozměry okolo $200\,\mathrm{mm}$. V průmyslu je pak rozměr dán potřebami využití kvadkoptéry. Obecně však platí, že čím větší kvadkoptéra, tím vyšší je její stabilita. Na druhou stranu příliš velké kvadkoptéry přináší i řadu nevýhod, jako potřebu užití větších (a nákladnějších) motorů, na což navazuje větší odběr proudu, tedy i potřeba pořízení baterie s vyšší kapacitou, která je obvykle těžší než baterie s kapacitou menší.

Vrtule

Vrtule je zařízení, které přeměňuje energii rotace na tah. Vrtule u multikoptér rozlišujeme na pravotočivé a levotočivé. U kvadkoptéry v konfiguraci X používáme stejný typ vrtule diagonálně, to znamená, že vrtule č. 1 a 3 jsou levotočivé a vrtule č. 2 a 4 jsou pravotočivé nebo naopak (číslování uvažujeme podle obr. 1.4). Dále se vrtule dělí podle počtu listů. Multikoptéry většinou využívají vrtule dvoulisté, případně třílisté. Důležitým parametrem je taktéž rozměr vrtulí. Při výběru rozměru musíme brát v úvahu velikost rámu a také výkon motoru tak, aby vrtule byly dostatečně velké a poskytovali potřebný tah, ale na druhou stranu aby nebyly příliš velké, což by mělo za následek to, že motory by nemusely tak velké vrtule unést a to by mělo za následek příliš velký odběr proudu. Vrtule jsou dostupné v různých materiálech od dřevěných, přes plastové až po karbonové. U multikoptér se nejvíce využívá plastových nebo karbonových vrtulí.

Motory

V současné době se pro multikoptéry využívají takzvané brushless střídavé motory (obr. 1.3), tedy motory s rotačním pláštěm. Účinnost motoru s rotačním pláštěm je obvykle 85-90%, kdežto motor s pevným pláštěm má účinnost obvykle 75-80%. Rozdíl v účinnosti znamená, že více energie motoru se mění na rotační sílu a méně energie se mění v tepelné ztráty.

Motory vybíráme také podle jejich parametrů. Můžeme se setkat s označením $1806\,/\,2300\,\mathrm{kV}$ nebo $2204\,/\,2100\,\mathrm{kV}$. První číslo v označení (např. 2204) nám udává velikost motoru. V tomto případě nám číslo 2204 říká, že průměr statoru je $22\,\mathrm{mm}$ a výška statoru je $4\,\mathrm{mm}$. Obecně pak můžeme říct, že čím větší motor (hlavně průměr statoru), tím větší kroutící moment bude motor mít. Vysoký kroutící moment je nezbytný pro nasazení větších vrtulí. Druhý parametr (například $2100\,\mathrm{kV}$) nám udává, kolik tisíc otáček za minutu na volt umí motor vyprodukovat. Místo $2100\,\mathrm{kV}$ se můžeme také setkat s označením

Obrázek 1.3: Brushless motor (zdroj: *hobbyking.com*)

2100 ot./min/V. Můžeme říct, že čím větší hodnota kV, tím menší vrtule můžeme použít. Je důležité tyto dva parametry vhodně zkombinovat s rozměry vrtulí.

Regulátory (ESC)

Regulátory (můžeme se setkat se zkratkou ESC) využíváme, jak již název napovídá, k regulaci otáček motoru. Pro každý motor je používán samostatný regulátor, takže například u kvadkoptéry musíme použít čtyři regulátory. Hlavním parametrem pro výběr vhodného regulátoru je velikost proudu, se kterým je regulátor schopen zacházet. Výběr regulátoru tedy úzce souvisí s výběrem motoru. Musíme se ujistit, že maximální proud, který regulátor zvládne je větší, než maximální odběr proudu (někdy značeno jako maximální zatížitelnost) motoru. Teoreticky bychom mohli použít 25 A regulátor pro motor s maximální zatížitelností 25 A, ovšem riskujeme přílišné zahřívání regulátorů a tím i destrukcí vlivem tepla.

Řídící jednotka

Řídící jednotka je „mozek“ celé multikoptéry, která se stará o stabilitu celého stroje. Ke své činnosti využívají digitální gyroskopy a akcelerometry. Některé řídící jednotky mají možnosti využití telemetrických dat či GPS systému pro funkce, jako například „return to home“ (návrat domů) při ztrátě signálu mezi strojem a vysílačem. Řídící jednotky se následně dají různě přeprogramovat tak, aby vyhovovaly požadavkům pilota. Například pro akrobacii se do řídící jednotky nahraje speciální „acro“ firmware, který zaručí citlivější ovládání, s nímž stabilizace modelu nebude tak přísná – vhodné pro akrobatické prvky, jako přemety.

Přijímač

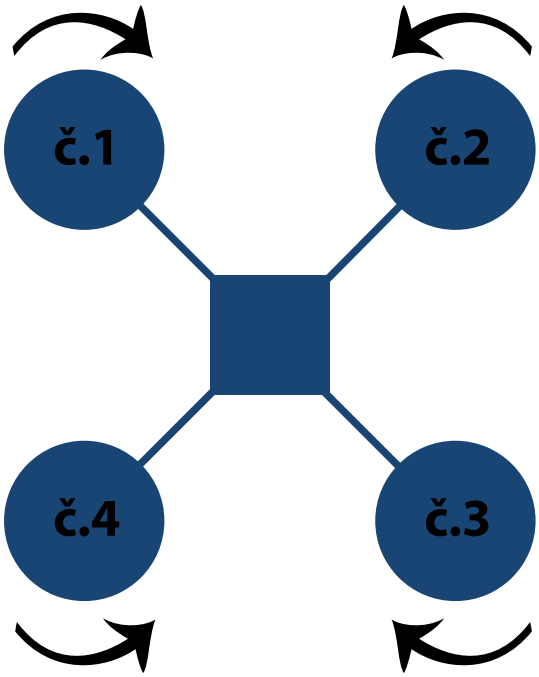
Aby mohla multikoptéra přijímat vysílaný signál z vysílače, musí mít na své straně příslušný přijímač. Přijímače disponují několika vlastnostmi, kterými se od sebe vzájemně odlišují – typ používané modulace, počet kanálů (tj. počet signálů, které dokáže přijímač rozlišit), pracovní frekvenční pásmo a dále pak rozměry a hmotnost. Pro správnou

funkci musí být přijímač s vysílačem spárován (někdy se tomuto procesu říká bindování - z anglického slova bind), což závisí na typu přijímače a vysílače.

Baterie

V současné době se využívá lithium polymerových baterií (LiPo), které dodávají elektrickou energii všem elektricky závislým komponentám. LiPo baterie mají schopnost uchovávat poměrně velké množství energie a také jsou také schopny tuto energii velmi rychle vydávat (čtyři motory na kvadkoptéře jsou velmi energeticky náročné). LiPo baterie mají několik parametrů, kterými se musíme při jejich výběru řídit. Prvním je počet článků baterie (značení např. 3S pro tříčlánkovou baterii), dalším potom kapacita baterie. Můžeme nabýt dojmu, že vyšší kapacita sebou přináší maximální výhodu-množství energie. Tento přínos je však „vykoupen“ rozměry a váhou baterie. Neméně důležitým parametrem je takzvaný C parametr, který udává maximální proudové zatížení (například 25C) a v neposlední řadě hodnota napětí, pod kterou baterie pracuje.

Na obrázku č. 1.4 je znázorněno, jakým směrem se vrtule točí u kvadkoptéry. Směr otáčení může být i opačný, ovšem musí být zachováno pravidlo otáčení do kříže – tedy motory na stejné diagonále se musí otáčet stejným směrem. Tato technika otáček nám zaručí stabilitu a ovladatelnost stroje. Samozřejmě je nutné pro otáčení ve směru hodinových ručiček použít vrtule pravotočivé a naopak při otáčení v protisměru hodinových ručiček použít vrtule levotočivé.



Obrázek 1.4: Směr otáčení vrtulí kvadkoptéry

1.3 Komunikace vysílač - přijímač

Z pohledu principu funkce dronů se v této práci budeme nejvíce věnovat komunikaci mezi vysílačem a dronem. Jak jsem již nastínil v předchozí kapitole č. 1.2, pro komunikaci mezi vysílačem a dronem je třeba využít komponent vysílače a přijímače.

Současné RC modely komunikují s vysílači v pásmu $2{,}4\,\mathrm{GHz}$, případně $5\,\mathrm{GHz}$, což přináší spoustu výhod, ale ne vždy tomu tak bylo. Ještě před několika lety měl každý modelář k dispozici sadu krystalů, které mu určovaly na jakém kanálu bude komunikace mezi vysílačem a modelem probíhat. Pokud měli dva modeláři krystal na stejném kanálu, docházelo k jejich významnému vzájemnému rušení.

Komunikace postupně přešla do pásma $2{,}4\,\mathrm{GHz}$, využívajícího techniku rozprostřeného spektra. Využití frekvenčních pásem $27\,\mathrm{MHz}$, $35\,\mathrm{MHz}$, $40\,\mathrm{MHz}$, $72\,\mathrm{MHz}$ a $75\,\mathrm{MHz}$ (příslušné kanály v těchto pásmech popisuje [7]), se snížilo.

1.3.1 Techniky rozprostřeného spektra

Metoda rozprostřeného spektra nám zajišťuje odolnost vůči rušení. Principem je využívání pro komunikaci záměrně širšího frekvenčního pásma, než je pro komunikaci nezbytně nutné – dochází k rozprostření signálu do širokého pásma.

Mezi nejrozšířenější techniky rozprostřeného spektra patří FHSS a DSSS.

FHSS

Frequency Hopping Spread Spectrum, neboli přeskakování kmitočtů v rozprostřeném spektru. Principem této techniky je přeskakování nosného signálu s namodulovanými daty mezi frekvencemi. K těmto přeskokům dochází ve velmi krátkých časových intervalech (maximálně $400\,\mathrm{ms}$). Vysílač i přijímač znají sekvenci těchto přeskoků a vždy se tedy včas přeladí na správný subkanál. Tato technika nevylučuje, že v některém okamžiku dojde k souběhu dvou signálů na stejném subkanálu, tj. dojde k rušení.

DSSS

Direct Sequence Spread Spectrum, neboli přímo rozprostřené spektrum. V této technice se předpokládá, že každý přenášený bit je nahrazen sekvencí bitů, pomocí Barkerova kódu. Tato sekvence se nazývá chip. Dochází tedy k redundanci bitů, tedy k rozprostření signálu do větší části frekvenčního spektra. Výsledkem je signál, který je méně citlivý na rušení. Na spektrálním analyzátoru takový signál vypadá jako šum.

Pro kontrolu přenesených dat mezi vysílačem a přijímačem je zaveden CRC kód (Cyclical Redundancy Code), pomocí něhož se kontroluje pravost dat. CRC kód nebude souhlasit, pokud alespoň jeden přijatý bit nesouhlasí s odeslaným. Klasické RC přijímače vadné pakety ignorují, modernější Wi-Fi přijímače si vyžádají poslání takového paketu znovu.

1.3.2 Technologie RC vysílačů

Současné RC vysílače většinou kombinují technologie rozprostřeného spektra. Například firma Futaba představila svůj 2,4 GHz vysílač s technologiemi FHSS a DSSS pod jednou (kombinovanou) technologií FASST. Firma Spektrum představila svůj vysílač pouze s technologií DSSS, pod názvem DSM. Později DSM rozšířili o dva frekvenční přeskoky pod názvem DSM2. U vysílače Spektrum s technologií DSMX byl pak počet přeskoků zvýšen ze dvou na dvacet tři [10].

V tabulce č. 1.1 jsou porovnány některé RC vysílače. Z tabulky je zřejmé, že všichni velcí výrobci RC vysílačů využívají obou technologií FHSS a DSSS.

Výrobce	**Marketingový název technologie**	**Šířka pásma DSSS**	**Počet subkanálů FHSS**
Futaba	FHSS	10 kHz	74
Futaba	FASST	1,6 MHz	36
Spectrum/JR	DSM	1 MHz	1
Spectrum/JR	DSM2	1 MHz	2
Spectrum/JR	DSMX	1 MHz	23
JR	DMSS	3 MHz	23
HiTec	AFHSS	1 MHz	20
Turnigy	V1	1 MHz	1
Turnigy	V2	1 MHz	16
Airtronics	FHSS-x	1 MHz	15

Tabulka 1.1: Přehled technologií RC vysílačů

V dnešní době je velmi populární u moderních dronů využívat přenos pomocí technologie Wi-Fi. Konkretní příklad a popis takového druhu komunikace je popsán v kapitole č. 4.2.1. V další kapitole se budeme věnovat automatickému rozpoznávání lidské řeči, technikám zaručujícím, že mluvené příkazy pro ovládání periferií dronu jsou vhodně a přesně zpracovány a takové příkazy pak odeslány do dronu pro další zpracování.

2 Lidská řeč a její automatické rozpoznávání

Lidská řeč je jednou z věcí, kterou se odlišujeme od ostatních živočichů a která nám dává jisté významné postavení v přírodě. Řeč je jakýsi nástroj pro převod našich myšlenek do verbální podoby a je tedy základním prvkem pro takzvané kolektivní vědomí. Schopnost mluvit a porozumět využíváme každým dnem, aniž bychom se nad tímto jevem pozastavovali. S pokročilou technologií již dnes můžeme pomocí lidské řeči ovládat i počítač. Ovšem abychom pochopili, jak ASR (z angl. Automatic Speech Recognition) software lidskou řeč rozpoznává a následně zpracovává, musíme si nejprve osvětlit několik termínů, týkajících se lidské řeči.

Artikulace

Artikulace je schopnost změny akustických vlastností hlasového traktu, v němž vzniká akustická vlna, která se z úst šíří volným prostorem k posluchači. Zdrojem energie akustické vlny je primárně výdech mluvčího. Artikulace se dělí na úseky znělé a neznělé. *Znělé* úseky vznikají tak, že proud vzduchu při průchodu hlasovým traktem prochází sevřenými hlasivkami, díky čemuž vznikají vibrace, které vytvářejí impulzy jdoucí do hrdelní, nosní a ústní dutiny. V těchto dutinách impulzy rezonují a vzniká tak znělá promluva, která je řízena svalovou činností rezonátorů (nosní, hrdelní a ústní dutiny). *Neznělé* úseky vznikají bez průchodu proudu vzduchu hlasivkami. Na obrázku č. 2.1 je znázorněn průchod proudu vzduchu a jeho přeměna na akustickou vlnu.

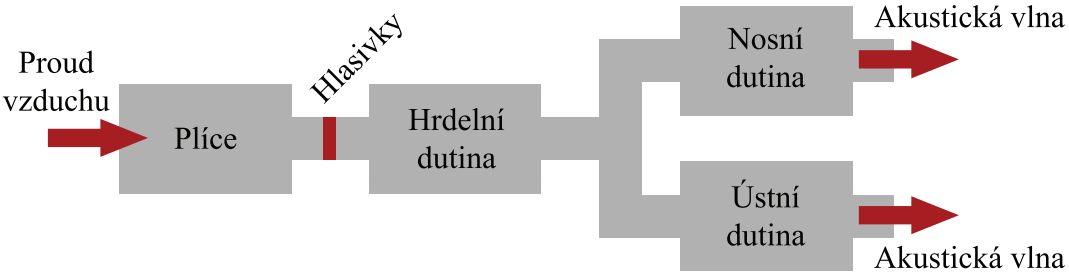
Percepce řeči

Zpracování akustické vlny posluchačem se nazývá percepce (vnímání) řeči. Percepce řeči je založena na mechanickém pohybu bubínků při průchodu akustickou vlnou. Pohyb bubínků vyvolává podráždění nervů vedoucích ze sluchového ústrojí do mozku, kde jsou tyto impulzy dále zpracovány.

Foném

Jedná se o nejmenší zvukovou jednotku, s jejíž pomocí lze mezi sebou rozlišovat jednotlivá slova. Foném, jako jednotka, nemá význam, vytváří ovšem vyšší, pochopitelné celky. Kombinací fonémů se utváří všechny výrazy daného jazyka. Například anglický jazyk obsahuje 44 fonémů, český jazyk 39 fonémů.

Automatické rozpoznávání řeči je důležitou technologií, která umožňuje a vylepšuje komunikaci ve vztahu člověk - člověk a také člověk - stroj. Technologie ASR je v aktivním vývoji po více než padesát let a za tuto dobu prošla mnohými změnami a vylepšeními. V minulosti ASR technologie nebyla považována za důležitou součást komunikace člověk-stroj, z části proto, že technologie té doby nebyly na takové úrovni, aby komunikaci člověk-stroj umožňoval a z



Obrázek 2.1: Vznik akustické vlny

části proto, že jiné interaktivní nástroje, jako klávesnice a myš, předčily hlasové povely svou rychlostí a přesností.

V posledních letech hlasové technologie mění způsob práce s některými zařízeními a stávají se nedílnou součástí našich životů a způsobu práce s novými technologiemi. Tento trend je posilován díky vyspělé výpočetní technice. Dnes jsou k dispozici multi-jádrové procesory a CPU/GPU clustery. Výkon počítačů je v současnosti několikanásobný v porovnání s jejich výkonem před deseti lety. Mohly se tak vyvinout rozsáhlejší a komplexnější ASR modely, které výrazně snižují chybovost ASR systémů. Navíc máme přístup k rozsáhlejšímu množství dat a to díky internetu a cloud computingu. Další věc, díky které jsou hlasové technologie stále populárnější, je používání moderních mobilních telefonů a takzvaných nositelných zařízení, jakými jsou například chytré hodinky. Hlasové technologie nacházejí uplatnění v inteligentních domech a také v automobilovém průmyslu. V této práci využíváme ASR systému k převodu hlasových příkazů do textových řetězců, které jsou dále zpracovávány a zasílány, jako řídící příkazy, do dronu.

Důležitou roli hrají ASR systémy na poli komunikace člověk-člověk (HHC – Human-Human Communication) a komunikace člověk-stroj (HMC – Human-Machine Communication).

Komunikace člověk-člověk

Pro překročení jazykových bariér mezi dvěma a více lidmi bylo v minulosti zapotřebí prostředníka – překladatele. V dnešní době, díky HHC systémům, tento prostředník odpadá a rozhovor mezi lidmi z různých koutů světa může probíhat jednoduše, díky překládání mluveného slova do jazyka, jemuž posluchač rozumí. Takovou funkcí disponuje například komunikační program Skype.

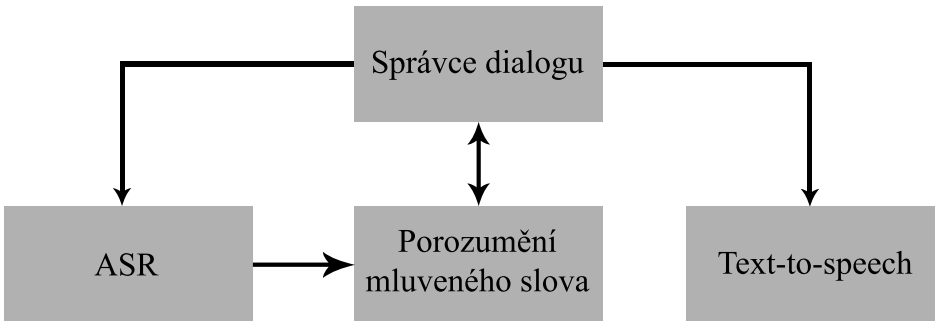
Další možnou aplikací je nahrávka zvukové stopy, její překlad, převedení do textu a následné odeslání tohoto textu e-mailem nebo SMS zprávou. Využití je možné například ve VoIP telefonii.

Komunikace člověk-stroj

ASR systémy jsou velmi důležitým prvkem HMC komunikace. Nejpopulárnějšími apli-

kacemi HMC komunikací jsou: hlasové vyhledávání, osobní asistenti, hlasové ovládání ve vozidlech nebo také ovládání inteligentních domů (ovládání termostatu, zamykání dveří, ...).

Tyto aplikace jsou příkladem takzvaného systému mluveného jazyka (spoken language system). Jak je vyobrazeno na obr. 2.2, systém mluveného jazyka sestává ze čtyř hlavních komponent: ASR komponenty, která převádí řeč na text; komponenty pro porozumění mluveného jazyka, která vyhledává relevantní informaci v mluvených slovech; komponenty text-to-speech, která zprostředkovává mluvené informace a komponenty správce dialogu, která komunikuje s ostatními třemi komponentami. V této práci se budeme zabývat zejména ASR komponentou [2].



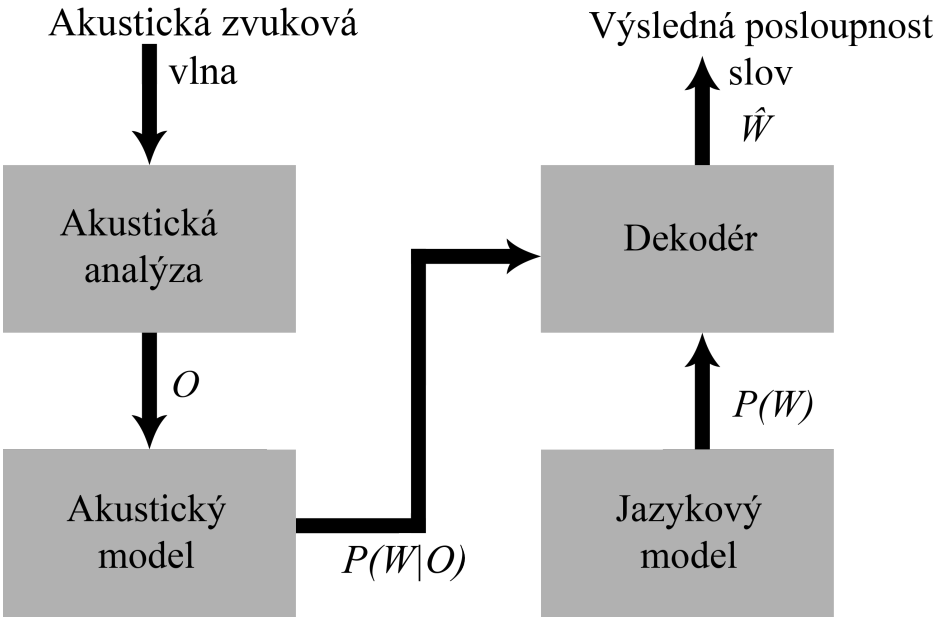
Obrázek 2.2: Systém mluveného jazyka

2.1 Základní architektura ASR systému

Na obr. 2.3 je vyobrazeno typické schéma architektury ASR systému. Architektura ASR systému se skládá ze čtyř základních komponent - komponenty akustické analýzy, akustického modelu, jazykového modelu a dekodéru.

Podle obrázku 2.3 můžeme funkčnost ASR systému rozdělit do několika dílčích kroků. Mějme posloupnost příznakových vektorů $O = \{o_1, o_2, \ldots, o_t\}$ a posloupnost slov $W = \{w_1, w_2, \ldots, w_n\}$, pak hledáme takovou posloupnost slov $\hat{W}$, kterou označíme jako nejpravděpodobnější posloupnost slov pro dané příznakové vektory. Výslednou posloupnost hledaných slov tak můžeme vidět na rovnici 2.1, která je argumentem maxima podmíněné pravděpodobnosti $P(W|O)$. Pravděpodobnost $P(W|O)$ lze podle Bayesovy věty rozložit na $P(W|O) = \frac{P(O|W)P(W)}{P(O)}$.

$$\hat{W} = argmax\, P(W|O) = argmax\, \frac{P(O|W)P(W)}{P(O)} \tag{2.1}$$



Obrázek 2.3: Architektura ASR systému

2.1.1 Akustická analýza

Vstupem komponenty akustické analýzy (někdy také komponenty pro extrakci příznaků) je akustická zvuková vlna, ze které je potlačen šum a redundantní informace z pohledu přenosu slovní informace. Jinými slovy má tato komponenta za úkol extrahovat zájmové oblasti řeči, tj. potlačení charakteristiky řečníka. Při této extrakci dochází k cílenému snížení objemu zpracovaných dat. Výstupem jsou příznakové vektory O, které určují zájmové prvky řeči řečníka bez redundantních vlastností řeči řečníka. V této fázi také dochází k převodu časové realizace signálu do frekvenční, pomocí Fourierovy transformace. Podobnou techniku převodu z časové oblasti do frekvenční můžeme vypozorovat i v lidském uchu. Řeč je totiž spojitá funkce změn akustického tlaku. Člověk tuto změnu tlaku neslyší, ale slyší frekvenci signálu (tóny). Tento převod se děje ve vnitřním uchu. Takové frekvenční spektrum je pak zpracováváno v mozku. Příznakové vektory O, které jsou výstupem akustické analýzy, jsou dále zpracovány akustickým modelem.

2.1.2 Akustický model

Komponenta akustického modelu integruje znalosti akustiky a fonetiky. Jeho vstupem je zájmový prvek z komponenty akustické analýzy. Úkolem akustického modelu je nalezení fonetických jednotek ve zpracovávaném signálu, tj. snaží se „naučit“, jak který foném zní. Pro naučení těchto fonémů využívá takzvaná řečová data, ve kterých je popsáno, jakému zvuku je přidružen příslušný foném. Po zkompletování těchto zvuků do znějícího slova se dále musí

rozlišovat mezi podobně znějícími slovy s jiným významem. Akustický model se stará o co nejpřesnější rozlišení mezi takovými slovy. V akustickém modelu se pracuje s kontextově závislými fonémy. Neutváří se tedy model pouze pro foném "A", ale i modely závislé na kontextu, tedy "X-A-Y", kde je X a Y předcházející, respektive následující foném. Ke své funkci využívá statistických modelů, mezi nejpoužívanější patří skrytý Markovův model (HMM), který je podrobněji popsán v kapitole č. 2.2. Dalším přístupem je metoda neuronových sítí, popsaná v kapitole 3.

2.1.3 Jazykový model

Jazykový model odhaduje pravděpodobnost správnosti určitého slova nebo sekvencí slov pomocí nauky o vztahu mezi slovy z jazykového korpusu. Přesnějšího výstupu jazykového modelu může být dosaženo předchozí znalostí požadavků či oblasti jazyka.

Jazykový korpus

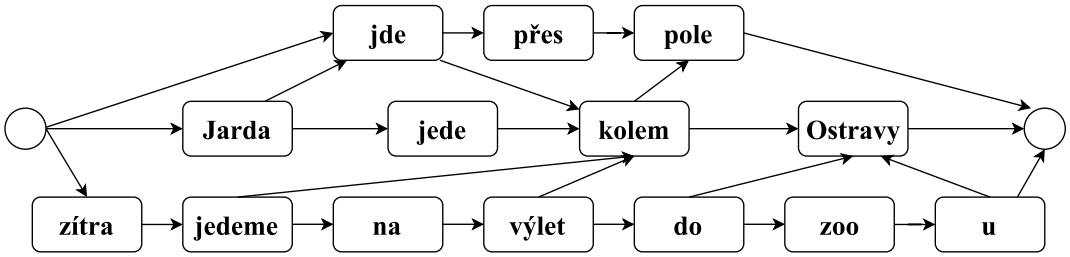
Jazykový korpus je rozsáhlý soubor textů daného jazyka, který je převeden do digitální podoby (pro lepší manipulaci a vyhledávání textů). Jazykového korpusu je využíváno především pro lingvistické výzkumy. V České republice se o jazykovým korpusem zabývá projekt Český národní korpus (ČNK) [14].

Problémem jazykového modelu, užívajícího hovorové řeči, je častá neplatnost gramatických pravidel. Řešením je použití statistického modelu, v němž se analyzují dvojice či trojice slov, pomocí kterých se následně vypočítává pravděpodobnost výskytu posloupnosti slov. Přirovnat to opět můžeme k lidskému vnímání, kdy i dítě, které neumí dostatečně dobře gramatiku, umí dobře mluvit. Je to dáno tím, že odposlechlo tyto vytvořené sekvence slov a ví (s určitou pravděpodobností), které slovo a v jaké podobě bude následovat na základě předchozích slov.

2.1.4 Dekodér

Cílem dekodéru je získat sekvenci slov $\hat{W}$ podle nejvyšší pravděpodobnosti rozpoznání, kombinací výstupů z jazykového a akustického modelu. Je nutné, aby algoritmus, který tuto sekvenci slov získá, byl co nejrychlejší, nejlépe v řádu milisekund. To je řešeno omezením prohledávané oblasti, pomocí prořezávání grafu, kdy se uvažují pouze pravděpodobné sekvence slov.

Výslednou gramatiku lze vizualizovat pomocí orientovaného grafu, jako na obrázku 2.4. Na tomto obrázku je vidět jednoduchý model gramatiky o několika frázích, který lze ještě dále rozšířit ohodnocenými hranami.



Obrázek 2.4: Gramatika jako orientovaný graf

2.2 Využívání skrytých Markovových modelů

Na skrytý Markovův model lze pohlížet jako na pravděpodobnostní konečný automat, který v čase generuje náhodnou posloupnost vektorů pozorování $\boldsymbol{O} = \{\boldsymbol{o_1}, \boldsymbol{o_2}, ...\boldsymbol{o_T}\}$. V každém kroku potom tento model změní svůj stav s_i podle předem daných pravděpodobnostních přechodů a_{ij}. Model poté přechází do stavu s_j, který generuje vektor pozorování $\boldsymbol{o_t}$ a to podle rozdělení výstupní pravděpodobnosti $b_j(\boldsymbol{o_t})$, příslušné k tomuto stavu [16].

Podle vzorce 2.2 lze říci, že pravděpodobnostní přechod a_{ij} určuje s jakou pravděpodobností přechází model ze stavu s_i do stavu s_j. Stav s_i je v kterémkoliv čase t a stav s_j je potom v čase $t + 1$.

$$a_{ij} = P(s_j = s(t+1)|s_i = s(t)) \tag{2.2}$$

Funkce výstupní pravděpodobnosti $b_j(\boldsymbol{o_t})$ popisuje pravděpodobnostní pozorování vektoru $\boldsymbol{o_t}$. Funkce $b_j(\boldsymbol{o_t})$ může reprezentovat pravděpodobnost v případě, kdy pozorování nabývá konečného počtu diskrétních hodnot. Naopak může tato funkce $b_j(\boldsymbol{o_t})$ reprezentovat hustotu pravděpodobnosti v případě, že jsou pozorování hodnotou spojité náhodné veličiny. Pro funkci $b_j(\boldsymbol{o_t})$ platí vzorec 2.3.

$$b_j(\boldsymbol{o_t}) = P(\boldsymbol{o_t}|s_j = s(t)) \tag{2.3}$$

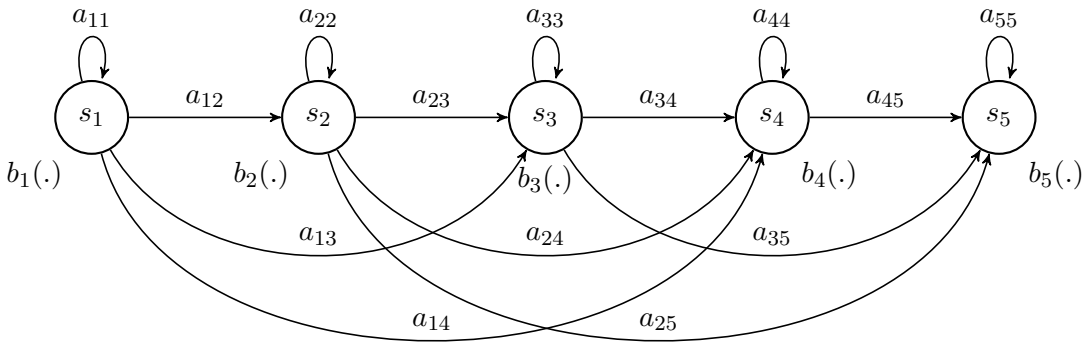
Tato výstupní pravděpodobnost musí být dostatečně konkrétní, aby bylo možné od sebe odlišit různé a zároveň i velmi podobné zvuky.

2.2.1 Konstrukce skrytých Markovových modelů

Jelikož je mluvená řeč procesem, který je postupem času dále vyvíjen, je vhodné využít takzvané levo-pravé Markovovy modely. Hlavním principem tohoto typu modelů je, že postupem času přechází z jednoho stavu do jiného, přesněji ze stavu s menším indexem do stavu s vyšším indexem, popřípadě stagnuje v aktuálním stavu. Nikdy tedy neprobíhá změna stavu z vyššího

do nižšího. Celý proces následně končí přechodem do stavu posledního, tedy do posledního spektrálního vzoru.

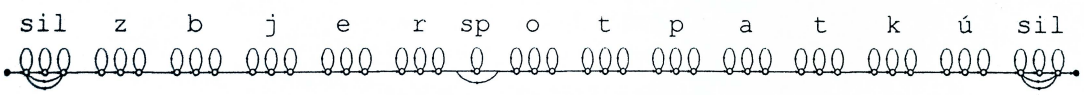
Takový levo-pravý skrytý Markovův model je vyobrazen na obrázku 2.5.



Obrázek 2.5: Pětistavový skrytý Markovův model

Pokud bychom uvažovali o práci s velkým množstvím (desetitisíce) slov, musíme brát v potaz, že trénování takového modelu, kde každé slovo vytváří jeden model, by bylo velice zdlouhavé a neefektivní. Je totiž třeba vytvořit hned několik trénovacích promluv pro každé slovo, aby se model přizpůsobil takovéto promluvě. Z tohoto důvodu je vhodné proces trénování skrytých Markovových modelů rozdělit do menších jednotek než jsou slova, tj. do fonémů.

Promluva, která je složena z fonémů, které reprezentují skryté Markovovy modely je vyobrazena na obrázku č. 2.6. Zde můžeme vidět, že každý foném je tvořen třístavovým modelem se třemi emitujícími stavy. Na začátku a na konci promluvy je dlouhá pauza, takzvaný *sil* (z anglického silence) a mezi slovy se vyskytuje krátká pauza *sp*, která je reprezentována jednostavovým modelem.



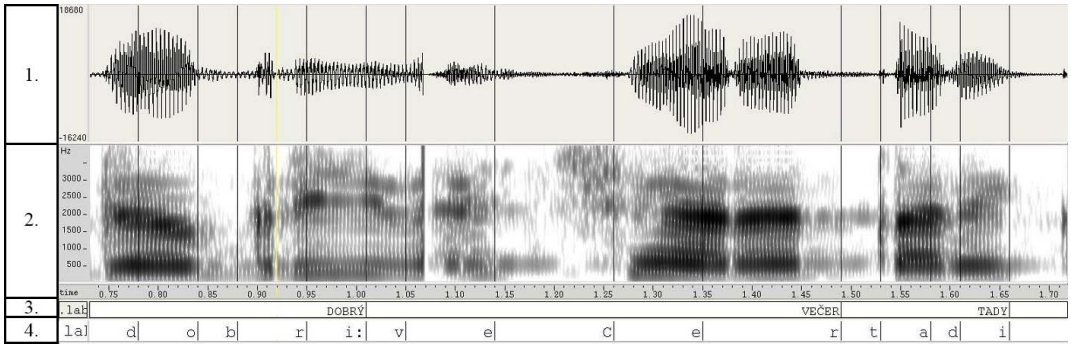
Obrázek 2.6: Promluva „Sběr odpadků“ reprezentována zřetězenými modely fonémů (Zdroj: Mluvíme s počítačem česky [16])

Používání modelů fonémů přináší do systémů rozpoznávání řeči mnoho výhod – rychlost systému a hlavně jeho rozšiřitelnost, to znamená, že můžeme doplňovat do slovníku nová slova, aniž bychom museli znovu trénovat promluvy s těmito novými slovy. Je ovšem nutné brát v potaz fakt, že modely pracující s celými slovy mají přesnější klasifikaci slov než modely se zřetězenými fonémy. Slova jsou zkrátka modelována s většími detaily [16].

2.3 Shrnutí ASR systémů

Cílem ASR je převod řeči do textu, kdy při této činnosti pracuje s akustickým a jazykovým modelem. Funkce ASR systému je založena na statistických modelech. ASR dokáže rozpoznat pouze slova, která předem zná (jsou obsažena ve slovníku). Nejlépe pracuje ve stejných akustických podmínkách, v nichž nedochází ke změně akustického kanálu či spontánnosti řečníka.

Na obrázku 2.7 je vyobrazena zvuková nahrávka z mikrofonu. První část obrázku obsahuje změny tlaku vzduchu v digitální podobě, tj. převod této změny do číselné podoby z rozsahu od -16240 do +18680. Cílem „výuky" ASR systémů je co nejpřesněji kopírovat funkci lidského sluchu. Jak je zmíněno v kapitole 2.1.1, lidský sluch slyší frekvence, nikoliv změny tlaku. Podobně, jako ve vnitřním uchu, i zde dochází k převodu číselné hodnoty akustického tlaku do frekvence, jak je vyobrazeno v druhé části obrázku 2.7. Hustota černé barvy v druhé části obrázku nám udává velikost zastoupení určité frekvence (tónu), tj. čím tmavší barva, tím hlasitější tón je vysloven. Třetí část obrázku nám ukazuje přepis mluveného textu v čase, v tomto případě tedy úryvek textu „Dobrý večer tady". Poslední, čtvrtá, část ukazuje fonetický přepis textu – každý symbol této části je foném.



Obrázek 2.7: Záznam zvukové nahrávky v digitální podobě (zdroj: Osel.cz [13])

Souhrnně můžeme tedy fungování ASR systémů sepsat následovně:

1. Mluvčí promlouvá k softwaru pomocí audio vstupu
2. Zařízení, ke kterému mluvčí promlouvá, vytváří vlnový soubor na základě jeho řeči
3. Vlnový soubor je následně pročištěn odstraněním okolního šumu a přizpůsobením hlasitosti
4. Výsledná vyfiltrovaná vlna je rozdělena na jednotlivé fonémy
5. Dále je analyzována sekvence fonémů. Software pak pomocí pravděpodobnostní analýzy utvoří z těchto fonému dané slovo

6. Software nyní „rozumí“ mluvčímu a může řetězec slov dále zpracovat, dle potřeb aplikace

Postupným nasazováním ASR systémů do současných, pokročilých aplikací se objevují nové překážky a výzvy. Každý člověk je obdařen rozličnými parametry hlasového ústrojí, projevujícími se například odlišnou barvou hlasu, přízvukem a v neposlední řadě také rozdílným tempem promluvy. Musíme brát v potaz také okolnosti, za jakých je rozpoznávání prováděno. Systém „naučený“ perfektně na jednoho řečníka nemusí fungovat bezchybně v případě, že je řečník například nachlazený nebo při promluvě šeptá. Faktorů, znemožňujících dokonalou funkčnost ASR systémů může být mnoho.

V následující kapitole se budu zabývat problematikou neuronových sítí, které udávají další přístup k realizaci akustického modelu, tj. pro potřeby určení správnosti vyslovených fonémů [1, 5, 16].

3 Umělé neuronové sítě

Matematické modely, vycházející z původních biologických neuronových sítí, při zachování hlavních funkcí a vlastností, nazýváme umělé neuronové sítě (ANN - z angl. Artificial Neural Network). Složeny jsou z prvků, takzvaných neuronů, které jsou vzájemně propojeny a mají schopnost učit se. Takto naučená neuronová síť potom dokáže podávat přesnější výsledky. Neuronové sítě jsou v současnosti využívány jako umělá inteligence například při rozpoznávání řeči či obrazů. Mezi konkrétní použití neuronových sítí můžeme zařadit například spamové filtry [11].

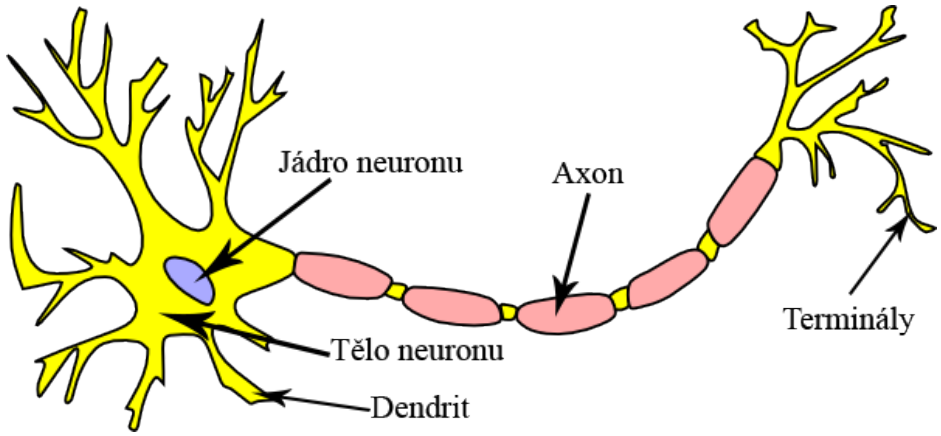
Trénování neuronových sítí sebou nese vysokou výpočetní náročnost. Vzhledem ke stále se zvyšujícímu výkonu výpočetních jednotek však v posledních letech jejich využití stoupá.

3.1 Neurony

Základním prvkem každé neuronové sítě, ať už biologické či umělé, jsou neurony. Tyto jsou navzájem propojeny a na jejich učení závisí přesnost výsledků neuronových sítí.

3.1.1 Biologické neurony

Biologický neuron, neboli nervová buňka, je základním prvkem nervové soustavy. Jsou to buňky určené k přenosu a uchování informací ke správnému fungování organismu jedince. Na obrázku 3.1 je vyobrazena struktura biologického neuronu.



Obrázek 3.1: Struktura biologického neuronu (zdroj: neuroscientificallychallenged.com)

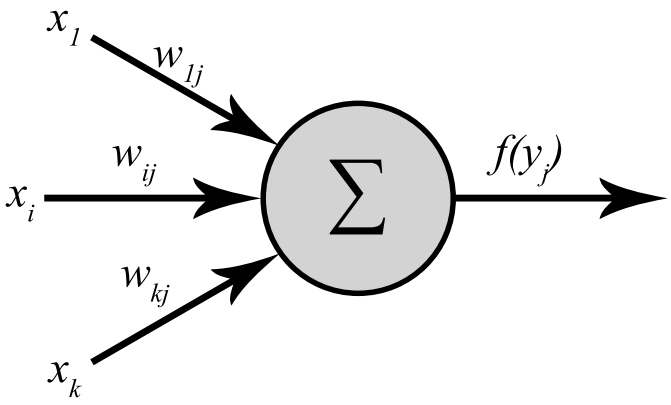
Neuron se skládá z jádra neuronu, těla neuronu, dendritů, axonů a axonové terminály. Základ každého neuronu je jeho tělo (soma), ve kterém je uloženo jádro. Axon a dendrity jsou přenosové kanály. Na konci axonu jsou terminály, které jsou v neuronové síti spojeny s dendrity jiných neuronů. Spojením neuronů vzniká neuronová síť, kterou se pomocí takzvané synapse

přenáší informace z jednoho neuronu do dalších. Samotný přenos probíhá tak, že tělo buňky a axony jsou zaobaleny membránou, která za určitých okolností vytváří elektrický impuls. Na dendrity jsou tyto impulsy přenášeny z axonu pomocí synaptických bran.

Propojení neuronů se v průběhu života mění – při zapomínání se synaptické spoje mezi neurony přerušují, při učení se naopak vytváří nové paměťové kanály.

3.1.2 Formální neurony

Základními stavebními prvky umělých neuronových sítí, jak je již zmíněno v úvodu kapitoly 3, jsou takzvané formální neurony. I když jsou to jednotky matematického modelu neuronové sítě, jejich vlastnosti a funkce jsou shodné s biologickými neurony.



Obrázek 3.2: Struktura formálního neuronu

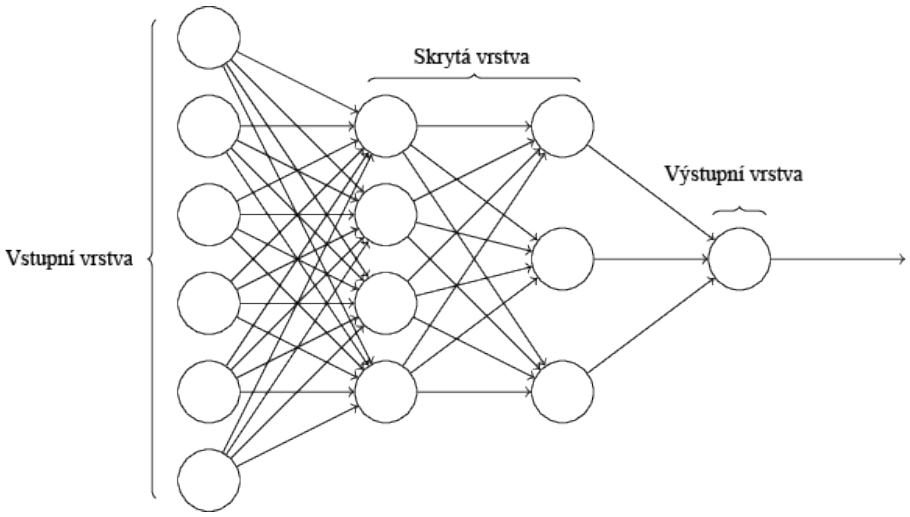
Podle obrázku 3.2 můžeme říci, že neuron $\sum$ má k vstupů, které modelují dendrity a udávají vstupní vektor $x = \{x_1, \ldots, x_k\}$. Vstupy jsou ohodnoceny takzvanými synaptickými váhami, které tvoří vektor $w = \{w_1 j, \ldots, w_{kj}\}$. Výstup neuronu $\sum$ je získán jako funkce $f(y_j)$. Vnitřní potenciál y_j, neuronu $\sum$ je definován jako suma vstupních hodnot y_j (vzorec 3.1).

$$y_j = \sum_{i=1}^{k} w_{ij} x_i \tag{3.1}$$

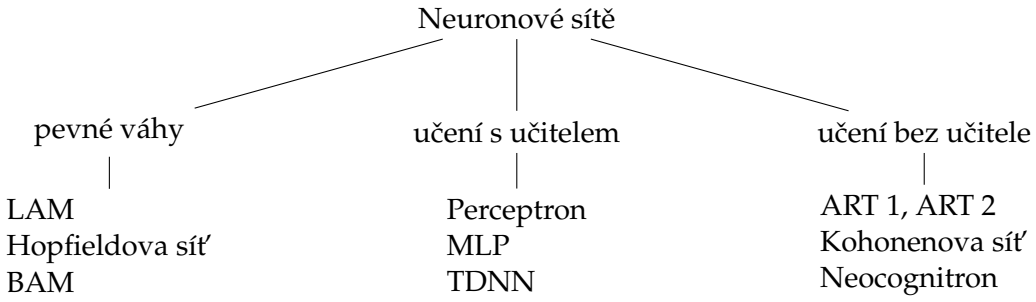
3.2 Struktura neuronových sítí

Spojením většího množství neuronů vzniká neuronová síť, jejímž hlavním přínosem je větší výpočetní sílu. Příklad neuronové sítě je vyobrazen na obrázku 3.3.

Na obrázku 3.3 je vidět čtyřvrstvá neuronová síť – část vlevo je vstupní vrstva, v níž jsou vstupní neurony. Část nejvíce vpravo je výstupní vrstva s výstupními neurony (v tomto případě je zde pouze jeden výstupní neuron). Ve střední části se nachází skryté vrstvy. Jde o vrstvy, které nejsou vstupní, ani výstupní. Sítím, které mají alespoň jednu skrytou vrstvu, říkáme multilayer perceptron (MLP).



Obrázek 3.3: Struktura neuronové sítě (zdroj: neuralnetworksanddeeplearning.com)



Obrázek 3.4: Typy neuronových sítí podle způsobu učení

Neuronové sítě se dále dělí podle typu učení (tématu učení neuronových sítí je věnována kapitola 3.3). Z obrázku 3.4 je patrné, že neuronové sítě dělíme podle učení do tří hlavních skupin a to na neuronové sítě s pevnými vahami, na sítě učící se s učitelem a na sítě učící se bez učitele.

Dalším kritériem pro rozdělení neuronových sítí je dělení dle struktury sítě. Máme tři druhy struktur neuronových sítí a to:

1. Jednovrstvé neuronové sítě
2. Vícevrstvé neuronové sítě (například obr 3.3)
3. Rekurentní neuronové sítě

3.3 Trénování umělých neuronových sítí

Cílem učení neuronových sítí je získání takového výstupu neuronové sítě, který odpovídá nastaveným vahám vstupních neuronů v síti. Učení neuronových sítí rozlišujeme na učení s uči-

telem a bez učitele.

Učení s učitelem

Tento typ učení je založen na zpětné vazbě. Výsledek vycházející z aktuálního nastavení je porovnáván s požadovaným výsledkem. Pomocí vypočtené chyby je vypočtena korekce, pomocí které je jsou upraveny hodnoty vah, za cílem snížit chybu. Poté se provede výpočet nového výsledku s upraveným nastavením vah a opět je vypočítána chyba a hodnota korekce vah. Vše se opakuje dokud není dosaženo minimální chyby.

Učení bez učitele

Při tomto typu učení není znám výsledek výstupu. Vstupem je sada setříděných vzorů podle určitých pravidel.

V následující kapitole se budeme zabývat praktickou implementací a nasazením ASR systémů pro potřeby hlasového ovládání dronů. Budou zde představeny stávající projekty hlasově ovládaných dronů a také popis popis komunikačního řetězce mezi dronem Parrot Bebop 2 a vysílačem SkyController.

4 Hlasové ovládání bezpilotních modelů

Praktickým výstupem této diplomové práce je aplikace pro mobilní zařízení, s jejíž pomocí může pilot dronu ovládat periferie modelu (konkrétně kameru) prostřednictvím hlasu. Drony, určené k průmyslovému využití, mají většinou kameru ovládanou pomocí dalšího vysílače. Je tedy zapotřebí dvou osob pro pilotování dronu – jeden člověk pilotuje dron, druhý člověk ovládá kameru (nebo gimbal). Výhodou použití aplikace pro hlasové ovládání periferií dronu je možnost ovládat dron, včetně periferií, pouze jednou osobou.

4.1 Stávající řešení dronů ovládaných hlasem

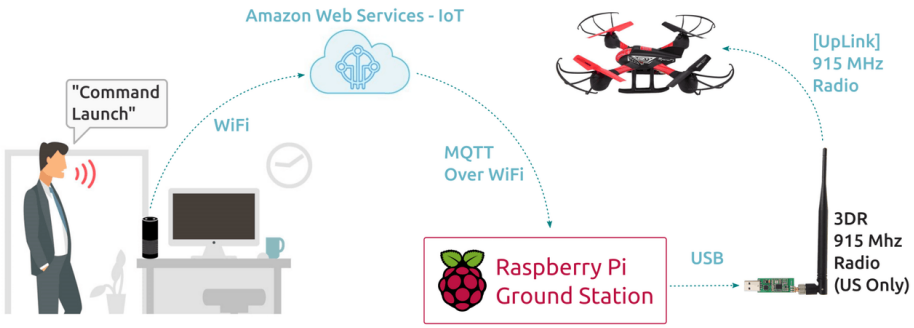
Drony ovládané hlasem jsou na současném trhu určeny spíše k hobby účelům. V průmyslové sféře se drony teprve zabydlují a zatím nebyly vyvinuty takové, které mají periferie ovládané jinak, než klasickým vysílačem. Za zmínku stojí, že příkazy pro hlasové ovládání dronů nejsou určeny k ovládání periferií, ale k ovládání samotného dronu, tj. k fyzickému pohybu dronu za pomoci jednoduchých hlasových příkazů.

Řešení této diplomové práce jde tedy trochu jinou cestou a to z praktických i legislativních důvodů – ovládat fyzický pohyb dronu hlasem je poněkud těžkopádné. Jinými slovy, pilot má větší cit v ovládání dronu pomocí rukou, hlavně díky rychlejší odezvě. Tím pádem by takovéto řešení nemuselo být legislativně schváleno při použití dronů v průmyslu.

4.1.1 Použité technologie hlasově ovládaných dronů

Hlasem ovládaný dron IRIS+

Projekt, který vyhrál *The AWS IoT Mega Contest*, využívá ke své funkci Raspberry Pi 2, model B a Amazon Alexa Echo, což je zařízení určené k hlasovému ovládání domácích spotřebičů. Softwarová část je pak zpracována pomocí Amazon Web Services AWS IoT – platformy pro komunikaci se zařízeními v internetu věcí (IoT).



Obrázek 4.1: Architektura hlasově ovládaného dronu IRIS+ (zdroj: hackster.io)

Amazon Alexa Echo má za úkol zpracovat hlasové příkazy, které jsou přenášeny formou textových řetězců do AWS IoT. Odtud jsou příkazy prostřednictvím komunikačního standardu MQTT přenášeny do Raspberry Pi, který v tomto případě funguje jako prostředník mezi AWS IoT a dronem. Rádiový modul 3DR, zapojený v Raspberry Pi pomocí USB portu, a který pracuje na frekvenci 915 MHz, v tomto případě funguje jako vysílač, který zasílá již zpracované hlasové příkazy do dronu ve formě příkazů, kterým dron rozumí. Na obrázku č. 4.1 je vyobrazena architektura tohoto řešení. Nevýhodou uvedeného řešení je poměrně dlouhá prodleva mezi vyslovením příkazu a finálním vykonáním příkazu dronem a také potřeba připojení prvků Amazon Alexa Echo a Raspberry Pi k internetu (částečně tento problém řeší funkce Device Shadows v AWS IoT, která využívá poslední známé konfigurace, pokud je zařízení offline), což je při práci v terénu nepraktické [19].

Hlasem ovládaný Bebop dron

Projekt společnosti Mobiquity Inc. je podobný projektu, který byl popsán v předcházejících odstavcích. Opět se využívá IoT platformy od Amazonu AWS IoT. Je zde ovšem využito jiné zařízení pro vstup hlasových povelů a to zařízení Amazon Echo Dot (druhá generace). Řízený dron Parrot Bebop je ovládán pomocí příkazů zasílaných z Raspberry Pi prostřednictvím node.JS knihovny `node-bebop` [1]. Tento projekt částečně řeší imobilitu, vzhledem k potřebě být připojen na domácí WiFi síti. Za pomoci linuxového démona `SystemD` jsou vytvořeny dva síťové stavy – prvním stavem je Raspberry Pi, vystupující jako hotspot pro konfiguraci známých sítí v případě nenalezení známých sítí; při druhém stavu se pak Raspberry Pi připojí na dvě různé WiFi sítě, zajišťující internetovou konektivitu [20].

Parrot AR.Drone 2 ovládaný za použití Intel RealSense SDK

Základem tohoto projektu, vyvíjeného dvojicí Marco Dal Pino a Marco Minerva, je komunikace s dronem AR.Drone 2, pomocí knihovny psané v programovacím jazyce C#. Pro rozpoznávání hlasu a převod na hlasové příkazy je použito programové vybavení firmy Intel RealSense SDK. Příkazy zasílané do dronu jsou v podobě AT příkazů, které umožňují základní operace s dronem jako vzlet, přistání a létání v různých směrech. Pomocí AT příkazů je také možné získat stream fotografií v reálném čase. Jelikož si dron AR.Drone 2 vytváří svou WiFi síť, je potřeba se na tuto síť připojit a dále posílat požadované příkazy z PC či z telefonu na IP adresu 192.168.1.1, skrze UDP na port 5556. Příkladem takových AT příkazů může být příkaz pro vzlet dronu – `AT * REF`. RealSense SDK pak poskytuje vcelku jednoduchý přístup k rozpoznávání hlasu za pomoci předdefinovaného seznamu klíčových slov (příkazů). Pro tyto účely bylo vytvořeno pole řetězců s jednoduchými příkazy: `Takeoff`, `Land`, `Rotate Left`, `Rotate Right`,

[1] Více informací o knihovně zde: `https://github.com/hybridgroup/node-bebop`

`Advance, Back, Up, Down, Left, Right, Stop, Dance`. Tento přístup k hlasově ovládaným dronům je, dle mého názoru, přímočařejší a praktičtější než dva výše zmíněné projekty. Není zde zapotřebí žádná interakce se servery přístupné skrze internet, tudíž se jedná o vhodný projekt i do terénu. Úryvky kódů je možné nalézt v [21].

Dron ovládaný hlasem založený na platformě Arduino
Odlišnost uvedeného projektu spočívá v použití vlastnoručně sestaveného dronu (trikoptéry), jehož stavbu autor v článku detailně popisuje. Cílem projektu je ovládání dronu mobilním telefonem, přes bluetooth sériovou linku. Řídící část je vykonávaná na desce Arduino Pro Mini s mikroprocesorem ATmega328. Část rozpoznávání hlasu probíhá na mobilním telefonu se systémem Android, který hlasové příkazy převádí na textové řetězce a ty následně zasílá přes bluetooth do Arduina. Zde se postupným přičítáním posílaných znaků vytvoří požadovaný řetězec s příkazem (výpis č. 4.1).

```
1 void loop() {
2   char c = Serial.read(); //Cteni znaku ze seriove linky
3   if (c == '#')
4       break;      //Pokud slovo konci znakem #, vyskoc ze smycky
5   voice += c;
6 }
```

Výpis 4.1: Úryvek kódu pro sestavení textového řetězce ze sériového toku

Řetězec `voice` z výpisu č. 4.1 je poté porovnáván s příkazy známými pro Arduino a pomocí funkce `analogWrite()` je na motor nebo motory zapsána analogová hodnota, která zapříčiní zvýšení či snížení tahu motoru či motorů. Tento projekt je určen pro hobby účely, pro létání na krátké vzdálenosti, vzhledem k použité technologii bluetooth (technologie PAN sítí). Zajímavostí je, že celý projekt, včetně stavby trikoptéry, údajně stál pouhých $11 [22].

4.2 Dron Parrot Bebop 2

Realizace hlasového ovládání periferií dronu probíhala na dronu Parrot Bebop 2, vyvinutém firmou Parrot, jehož lze ovládat klasickým vysílačem (pojmenovaným SkyController) nebo prostřednictvím mobilního telefonu s příslušnou aplikací, dostupnou pro platformy iOS a Android. Parrot Bebop 2 disponuje baterií o kapacitě 2700 mAh, díky které létá až 25 minut. Fotoaparát s rozlišením 14 Mpx je osazen širokoúhlým objektivem typu rybí oko a je schopný pořizovat video snímky ve Full HD kvalitě 1080p. Obraz je digitálně stabilizován, takže i při

náklonu dronu zůstává obraz na stejném místě a je stabilizován vůči vibracím. Videa a fotografie se ukládají do flash paměti dronu o kapacitě $8\,\mathrm{GB}$.

Konektivita je realizována pomocí standardu Wi-Fi 802.11a/b/g/n/ac s využitím multi-anténního systému MIMO. Dipólové antény pracují na frekvencích $2{,}4\,\mathrm{GHz}$ a $5\,\mathrm{GHz}$. Výstupní síla signálu je potom do 21 dBm. Uváděný dosah je $300\,\mathrm{m}$. Maximální horizontální rychlost dronu je $16\,\mathrm{m\,s^{-1}}$, maximální vertikální rychlost je potom $6\,\mathrm{m\,s^{-1}}$ [17].

4.2.1 Komunikace mezi vysílačem a dronem

Při realizaci hlasového ovládání periferií dronu využívám dron Parrot Bebop 2, popíšu tedy komunikaci mezi vysílačem a přijímačem konkrétně na tomto modelu. V předchozí kapitole 4.2 bylo zmíněno, že veškerý přenos dat mezi vysílačem a dronem je založen na Wi-Fi technologii. Při komunikaci se využívá protokolů a knihoven, které jsou obsaženy v ARSDK 3, tedy v SDK, které vývojáři usnadňují práci při vývoji aplikací pro různá Parrot zařízení.

V knihovnách ARSDK se můžeme setkat s označením „c2d“ nebo „d2c“, což znamená „controller to device“ nebo „device to controller“. Před studováním knihoven ARSDK je nutné uvědomit si, že Parrot produkty se v knihovnách značí jinými názvy, než pod jakými se prodávají. Konkrétně Parrot Bebop 2 je v knihovnách označován jako ARDrone nebo ARDrone 3; vysílač SkyController je většinou označen zkratkou SC.

4.2.1.1 ARDiscovery protokol

Knihovna obsahující ARDiscovery je rozdělena do dvou částí: Discovery a Conection. Část Discovery je zodpovědná za nalezení podporovaného zařízení ve Wi-Fi síti, zatímco část Conection zajišťuje sjednání parametrů pro spojení mezi vysílačem a dronem – tyto parametry pak využívají knihovny ARNetworkAL a ARNetwork, které realizují samotné zasílání dat.

Discovery

ARSDK využívá protokolu mDNS pro své propagování na Wi-Fi síť. Pomocí mDNS je možné vyčíst informace obsažené v tabulce č. 4.1.

Informace	Zdroj	Popis
Jméno	Název služby	Zobrazení jména produktu
IP	Rozlišení služby	IP adresa produktu
Port	Port služby	Discovery Conection port
Extras	TxtData služby	JSON data obsahující doplňující informace (v současné době obsahuje sériové číslo produktu)

Tabulka 4.1: Přehled informací získaných z mDNS

Conection

Podmínkou pro navázání spojení mezi vysílačem a dronem je další sjednání podmínek přenosu. To je zajištěno skrze TCP socket, kdy se využívá mDNS Service portu. Podmínky pro spojení jsou zasílány formou JSON řetězce, který obsahuje informace z tabulky č. 4.2.

Klíč	Povinnost	Popis
`d2c_port`	Ano	UDP port, který bude využit pro čtení dat
`controller_type`	Ano	Typ vysílače (např.: "Phone", "Tablet")
`controller_name`	Ano	Název vysílače (např.: Název aplikace)
`device_id`	Ne	Sériové číslo produktu

Tabulka 4.2: Dostupné klíče pro spojení

Příklad JSON řetězce:

```
{"d2c_port":43210, "controller_type":"Phone",
"controller_name":"com.example.arsdkapp"}
```

Klíč `"device_id"` je užitečný v případě opětovného připojení k dronu. Pokud dron obdrží požadavek na spojení obsahující klíč `"device_id"`, je spojení navázáno pouze v případě shody se sériovým číslem dronu.

Zařízení pak odpoví na požadavek na spojení opět skrze TCP soket jiným JSON řetězcem, obsahujícím informace z tabulky č. 4.3.

Klíč	Povinnost	Popis
`status`	Ano	Pokud je hodnota jiná než 0, je spojení odmítnuto
`c2d_port`	Ano	UDP port, který bude využit pro zasílání dat
`arstream_fragment_size`	Ne	Velikost `ARStream` fragmentů
`arstream_fragment_maximum_number`	Ne	Maximální počet `ARStream` fragmentů na videosnímek
`arstream_max_ack_interval`	Ne	Maximální interval mezi `ARStream` ACK signály
`c2d_update_port`	Ano	FTP port pro aktualizování zařízení
`c2d_user_port`	Ne	FTP port pro jiné účely (definované vývojářem)
`skycontroller_version`	Pouze SC	Verze SkyControlleru

Tabulka 4.3: Dostupné klíče pro odpověď na požadavek na spojení

Příklad odpovědi na požadavek na spojení formou JSON řetězce:

```
{"status":0, "c2d_port":54321,
```

```
"arstream_fragment_size":65000,
"arstream_fragment_maximum_number":4,
"arstream_max_ack_interval":-1, "c2d_update_port":51,
"c2d_user_port":61}
```


4.2.1.2 ARNetworkAL a ARNetwork protokol

Knihovna ARNetworkAL je zodpovědná za síťovou abstrakci knihovny ARNetwork. Obě knihovny jsou úzce svázané a struktura síťového paketu je složena z hlavičky obsahující obě knihovny. Každé zařízení má svoji vlastní konfiguraci protokolu ARNetwork. Taková konfigurace určuje počet a typ použitých bufferů. ID čísla bufferů jsou zasílány v ARNetworkAL paketech.

Cílové porty pro přenos UDP datagramů jsou sjednávány ve fázi ARDiscovery.Conection. Rámec obsahuje následující informace:

- Datový typ (1 byte)
- ID cílového bufferu (1 byte)
- Sekvenční číslo (1 byte)
- Velikost rámce (4 byty)
- Data (N bytů)

Pole datového typu podporuje čtyři typy dat: Ack(1) – potvrzení předchozích přijatých dat; Data(2) – klasická data (ACK není vyžadováno); Low latency data(3) – s daty s nízkou odezvou je zacházeno jako s klasickými daty, ovšem dostávají vyšší prioritu, tzn. jsou zpracovány dříve; Data s ACK(4) – Data, která vyžadují potvrzování – příjemce musí zasílat ACK zpět odesílateli.

Pole ID bufferu je rozděleno do tří sekci:

- [0; 9]: Rezervováno pro vnitřní užití knihovny ARNetwork
- [10; 127]: Datové buffery
- [128; 255]: Potvrzovací ACK buffery

Dle konvence je vhodné datové buffery používat tak, aby při „c2d“ začínaly na pozici 10 a stoupaly, a při „d2c“ začínaly na pozici 127 a klesaly.

Pole sekvenčního čísla značí nezávislé sekvenční číslo každého bufferu, které se s novými daty inkrementuje.

4.2.1.3 ARCommand protokol

Knihovna ARCommand je určená k implementaci kodeku, který má za úkol zasílání binárních dat mezi vysílačem a dronem.

Každý zasílaný příkaz (command) je identifikován prvními čtyřmi byty:

- ID projektu (1 byte)
- ID třídy v projektu (1 byte)
- ID příkazu ve třídě (2 byty)

Podle těchto ID je definována notace `Projekt.Třída.Příkaz`. Příkladem může být zaslání takzvaného TakeOff příkazu do dronu, tedy příkazu zajišťující jeho vzlet. Podle výše uvedené notace by takový příkaz byl `ARDrone3.Piloting.TakeOff`. V jazyce Java by se takový příkaz provedl podle výpisu č. 4.2.

```
1 import com.parrot.arsdk.arcommands;
2 import com.parrot.arsdk.arcontroller;
3
4 public class BebopDrone {
5    private ARDeviceController mDeviceController;
6    public void takeOff() {
7          mDeviceController.getFeatureARDrone3().sendPilotingTakeOff();
8     }
9 }
```

Výpis 4.2: Zjednodušená ukázka implementace metody takeOff()

Všechny příkazy jsou definovány v XML souborech a veškeré zdrojové kódy jsou generovány pomocí těchto XML souborů při kompilaci. XML soubory je možné nalézt v SDK adresáři `<SDK>/libARCommand/Xml`, v němž každý XML soubor náleží určitému projektu. Je nutné myslet na to, že každý Parrot produkt umí a využívá pouze určité funkce a tudíž se zdrojové kódy generují z rozdílných XML souborů, avšak z jednoho globálního SDK.

4.3 Rozpoznávání hlasu pomocí CMU Sphinx

V této podkapitole se budu zabývat nástrojem pro rozpoznávání hlasu CMU Sphinx. Jedná se o sadu knihoven a nástrojů, s jejichž pomocí lze vytvářet aplikace schopné zpracovávat textové řetězce vytvořené rozpoznáním hlasu. K nalezení nejvhodnější cesty přes akustický a jazykový model využívá skrytých Markovových modelů. Projekt CMU Sphinx byl vytvořen skupinou

The Sphinx Group na Carnegie-Mellonově univerzitě v Pensylvánii. Jedná se o open source projekt, jehož vývoj započal již v devadesátých letech dvacátého století.

CMU Sphinx disponuje několika různými balíčky, pro různé úlohy a aplikace:

- Pocketsphinx – odlehčená verze rozpoznávače hlasu napsaného v jazyce C
- Sphinxbase – podpůrná knihovna využívána Pocketsphinx
- Sphinx4 – nastavitelný a modifikovatelný rozpoznávač hlasu napsaný v jazyce Java
- Sphinxtrain – nástroje pro trénování akustických modelů

Pro své účely jsem se rozhodl využít balíček Pocketsphinx.

4.3.1 Vytvoření jazykového modelu

Existuje několik modelů popisujících rozpoznávaný jazyk. Patří mezi ně rozpoznávání pomocí gramatiky, statistických jazykových modelů, fonetických statistických jazykových modelů a seznamy klíčových slov nebo frází.

Pro účely této diplomové práce se nejvíce hodí rozpoznávání pomocí klíčových slov a frází. To znamená, že není třeba vytvářet rozsáhlou databázi slov, která by se rozpoznávala. Stačí vytvořit seznam klíčových slov a frází – příkazů, o nichž víme, že budou v aplikaci používána. To zajistí, že při rozpoznávání tímto typem modelu, bude rozpoznáno *pouze* definované slovo či fráze a ostatní slova, která jsou k rozpoznávači promlouvána, jsou ignorována.

Pokud bychom k vývoji aplikace přistoupili využitím rozsáhlého modelu gramatiky a tento jazykový soubor redukovali pouze na nám významná slova, vyvstal by problém v případě použití slov neobsažených v definovaném souboru. Rozpoznávač by vyvinul maximální snahu taková slova přiřadit ke slovům obsaženým v jazykovém souboru. Následkem této snahy je významná nepřesnost v rozpoznávání a náchylnost na okolní šum a nevýznamné promluvy.

Další výhodou používání seznamu klíčových slov a frází je možnost přiřazení „prahu významnosti“ každému klíčovému slovu či frázi, což znamená, že přiřazujeme přesnost, s níž má být dané klíčové slovo či fráze rozpoznáváno. To se hodí v případě, že máme dvě nebo více klíčových slov znějících podobně.

Pro kratší fáze by měl být práh významnosti nižší, například `1e-5`, pro delší fráze naopak vyšší, například `1e-30`. Tyto prahy významnosti je nutno vyladit tak, aby výsledek byl výsledkem co nejpřesnějším. Je vhodné používat klíčové fráze o třech až čtyřech slabikách. Kratší fráze se častěji pletou.

Seznam klíčových slov a frází, konkrétně u aplikace vycházející z této diplomové práce, je vyobrazen v tabulce č. 4.4.

Příkaz	Prah významnosti	Popis
`make a left move`	1e-10	Otáčení kamery doleva
`make a right move`	1e-10	Otáčení kamery doprava
`make an up move`	1e-10	Otáčení kamery nahoru
`make a down move`	1e-10	Otáčení kamery dolů
`cancel`	1e-10	Deaktivace naslouchání příkazů
`take a picture`	1e-30	Pořízení snímku z kamery
`stop`	1e-5	Zastavení otáčení kamery
`center`	1e-10	Vycentrování kamery
`record a video`	1e-50	Započetí nahrávání videa
`end recording`	1e-8	Zastavení nahrávání videa

Tabulka 4.4: Seznam klíčových slov a frází v souboru keywords.list

4.3.2 Fonetický slovník

Cílem fonetického slovníku je mapování slov ve slovníku na sekvenci fonémů. Slovník může obsahovat dvě a více slov s alternativní výslovností. Je nutné, aby tímto způsobem vytvořený slovník obsahoval všechna slova, která jsou aplikací používána. V opačném případě nebude rozpoznávač schopen promluvu identifikovat. CMU Sphinx již v základní verzi disponuje hotovými slovníky pro následující jazyky: anglický, francouzský, německý, ruský, holandský, italský, španělský a standardní čínský.

Ve výpisu č. 4.3 je ukázka několika slov z anglického slovníku `cmudict-en-us.dict`, který obsahuje přes 130 000 záznamů.

```
1 hello HH EH L OW
2 listen L IH S AH N
3 tomato T AH M EY T OW
4 tomato(2) T AH M AA T OW
```

Výpis 4.3: Ukázka definování slov ve slovníku

Na ukázce 4.3 je také vidět, na řádcích č. 2 a 3, různá výslovnost slova `tomato`, kdy další verze výslovnosti je definována číslem v závorce.

V následující kapitole se budu zabývat praktickou částí diplomové práce – tedy implementací aplikace pro hlasové ovládání periferií dronu Parrot Bebop 2.

5 Implementace aplikace Andron

Praktickým výstupem této diplomové práce je mobilní aplikace pro platformu Android, jejíž funkcí je zasílání hlasových příkazů do dronu Parrot Bebop 2. Úkolem není hlasem ovládaný fyzický pohyb dronu, nýbrž hlasem ovládaná kamera na něm umístěná. Aplikace, kterou jsem pojmenoval Andron, je podporována na OS Android ve verzi 4.2 a výše. Při implementaci aplikace jsem využil programového vybavení firmy Parrot `ARSDK`, zajišťující komunikaci mobilní aplikaci s dronem, respektive SkyControllerem. Popisem `ARSDK` se zabývám v kapitole č. 4.2. Pro implementaci rozpoznávače hlasu a definici hlasových příkazů jsem využil nástroje `CMU Sphinx`, popsaného v kapitole č. 4.3.

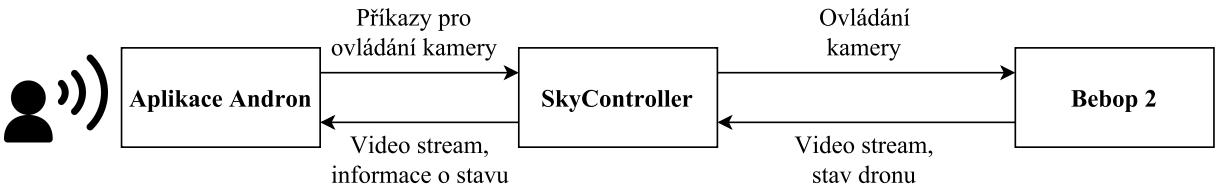
Implementované funkce tohoto řešení jsou:

- Ovládání otáčení kamery ve čtyřech směrech – nahoru, dolů, doprava, doleva
- Vycentrování kamery
- Pořízení fotografie z kamery
- Spuštění a ukončení nahrávání videa z kamery

Pro aktivaci hlasového rozpoznávání je třeba říci klíčové slovo `listen`, tím aplikace začne naslouchat hlasovým příkazům, určeným dronu. Pro zastavení otáčení kamery je třeba vyslovit klíčové slovo `stop`. Slovem `cancel` se naslouchání příkazů deaktivuje. Seznam klíčových slov a frází je sepsán v tabulce č. 4.4.

Všechny hlasové příkazy jsou pro větší univerzalitu implementovány v anglickém jazyce. Aplikace ovšem disponuje dvojjazyčným uživatelským rozhraním – českým a anglickým jazykem. Implementace aplikace proběhla v jazyce Java. Využívaná knihovna `Pocketsphinx` je psaná v jazyce C.

Na obrázku č. 5.1 je vyobrazeno blokové schéma projektu.



Obrázek 5.1: Blokové schéma projektu

Programová dokumentace je popsána v příloze B.

5.1 Struktura aplikace

Aplikace je pro lepší orientaci rozdělena do čtyř balíčků:

- `cz.beles.andron.activity`
- `cz.beles.andron.discovery`
- `cz.beles.andron.drone`
- `cz.beles.andron.view`

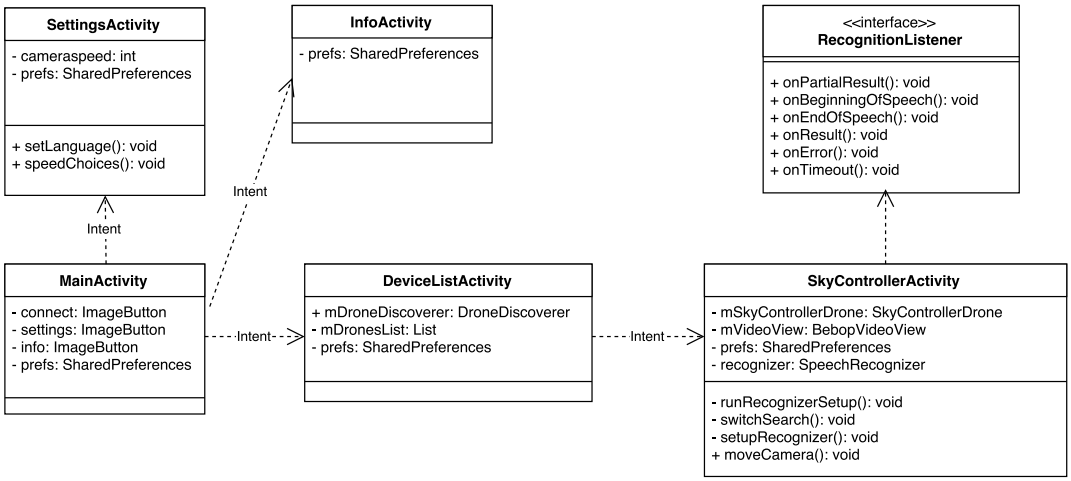
V balíčku `activity` jsou, jak název napovídá, obsaženy všechny použité aktivity aplikace (viz obrázek č. 5.2). Balíček `discovery` obsahuje třídu `DroneDiscovery`, která je určena k nalezení dostupných Parrot zařízení. K tomuto účelu se využívá funkcí obsažených v ARSDK. Balíček `drone` obsahuje třídu `SkyControllerDrone`, která se využívá pro zasílání již zpracovaných hlasových příkazů dronu. Balíček `view`, obsahující třídu `BebopVideoView`, má za úkol získání video streamu dronu Bebop 2. Tento stream se pak využívá v aktivitě `SkyControllerActivity`, konkrétně v jeho layoutu `activity_skycontroller.xml`.

Aplikace je složena z pěti aktivit (obrazovek), jak je patrné z diagramu na obrázku č. 5.2. Ne všechny aktivity jsou mezi sebou přímo propojeny. Konkrétní vazby mezi aktivitami jsou vyznačeny přerušovanou čarou se slovem `Intent`. Z diagramu vyplývá, že hlavní aktivitou aplikace je `MainActivity`, ze které pomocí tlačítek `connect`, `settings` a `info`, lze přistupovat k dalším aktivitám.

Aktivita `InfoActivity` má funkci informativní a pomocnou. Jedná se o aktivitu obsahující pouze text, který má uživateli pomoci s ovládáním aplikace.

Aktivita `SettingsActivity` je aktivitou pro nastavení aplikace. Obsahuje dvě položky: nastavení rychlosti otáčení kamery; nastavení jazyka aplikace.

Pomocí tlačítka `connect` se uživatel dostane do aktivity `DeviceListActivity`, která má za úkol vypsat do seznamu `droneList` výpis aktuálně podporovaných a připojitelných zařízení. Po výběru konkrétního zařízení se uživatel přesune do nejdůležitější aktivity `SkyControllerActivity`, ve které je vyobrazen video stream z kamery umístěné na dronu. Aktivita `SkyControllerActivity` implementuje rozhraní `RecognitionListener`, pomocí kterého se nastavuje funkčnost rozpoznávače hlasu.



Obrázek 5.2: Diagram znázorňující vztahy mezi aktivitami

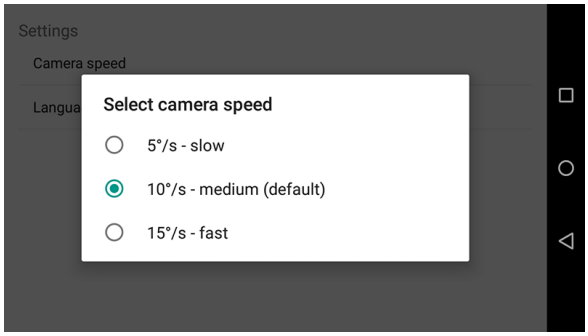
5.2 Návrh grafického rozhraní

Při grafickém návrhu rozhraní aplikace bylo cílem jednoduché, funkční rozhraní. Na obrázku č. 5.3 je vidět hlavní obrazovku s velkým tlačítkem pro připojení k zařízení. Velikost tlačítka značí jeho důležitost. V horních rozích jsou pak dvě další tlačítka: v levém horním rohu pro obrazovku s informacemi o aplikaci; v pravém horním rohu pro nastavení aplikace.

Nastavení aplikace je vyobrazeno na obrázku č. 5.4, kde je již vybrána položka s nastavením rychlosti otáčení kamery dronu. Jak je na tomto obrázku vidět, aplikace disponuje třemi rychlostmi otáčení obrazovky: 5°/s – pomalé otáčení; 10°/s – středně rychlé otáčení; 15°/s – rychlé otáčení. Nastavení se uchovává i po uzavření aplikace.

Na obrázku č. 5.5 je vyobrazena obrazovka se seznamem zařízení, která jsou schopna spárovat se s aplikací. Pokud není v dosahu nalezeno žádné zařízení, aplikace uživatele upozorní změnou rozvržení aktivity – místo seznamu `ListView` je zobrazen `RelativeLayout` s chybovým obrázkem a textem. Tento `RelativeLayout` má ve výchozím stavu nastaven parametr `android:visibility="gone"`, což znamená, že není na obrazovce vidět. Zobrazí se pouze při absenci zařízení k připojení. Toto je umožněno nastavením `setEmptyView()` na `ListView`, se seznamem zařízení [6].

Obrázek 5.3: Hlavní obrazovka

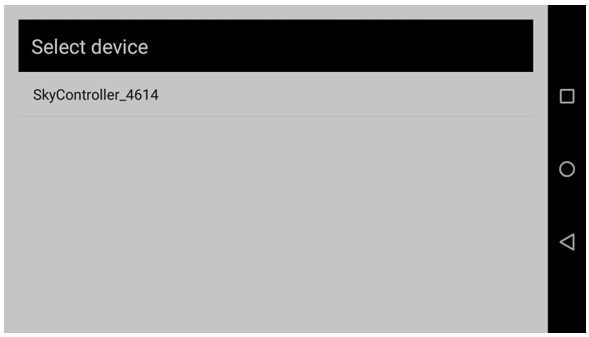


Obrázek 5.4: Obrazovka nastavení

Klíčový screen shot projektu je zachycen na obrázku č. 5.6. Na pozadí aktivity je zobrazen video stream z dronu v reálném čase a to pomocí vlastního view `videoView`, viz kapitola č. 5.5.

Na obrázku č. 5.6 je vidět v levém horním rohu obrazovky stav baterie vysílače SkyController a dronu Bebop 2. Ikona v pravém horním rohu značí stav nahrávání videa. Konkrétně v tomto případě se video nenahrává (ikona je bílá), nahrávání je vyznačeno změnou barvy ikony na červenou. V levém dolním rohu obrazovky je popsán stav rozpoznávače hlasu. V tomto konkrétním případě je rozpoznávač aktivní a čeká na hlasové příkazy, které přeposílá do SkyControlleru.

Při pořízení fotografie příkazem `take a picture` proběhne animace v podobě probliknutí bílého obdélníku přes celou obrazovku. Obdélník představuje komponentu `FrameLayout`, která je nad všemi ostatními prvky rozhraní. Má bílou barvu pozadí, viditelnost je nastavena na hodnotu `gone` – neviditelný. Při provedení příkazu `take a picture` se tento obdélník na okamžik zobrazí a proběhne animace snížení průhlednosti z hodnoty 1 na hodnotu 0 – úplná průhlednost. Celá animace trvá 300 ms.



Obrázek 5.5: Obrazovka seznamu zařízení pro připojení

Obrázek 5.6: Obrazovka s video streamem z dronu

5.3 Implementace rozpoznávače hlasu

Implementace rozpoznávače hlasu je realizována v aktivitě `SkyControllerActivity`, kde probíhá implementace rozhraní `RecognitionListener` z balíčku `Pocketsphinx`, jak bylo zmíněno v kapitole č. 5.1. Rozhraní `RecognitionListener` nám implementuje následující metody:

- `onPartialResult()` – kód uvnitř této metody se vykonává v průběhu rozpoznávání
- `onBeginningOfSpeech()` – kód uvnitř této metody se vykonává při započetí promluvy
- `onEndOfSpeech()` – kód uvnitř této metody se vykonává po ukončení promluvy
- `onResult()` – kód uvnitř této metody se vykonává, jakmile je rozpoznán celý řetězec mluveného slova (až po metodě `onEndOfSpeech()`)
- `onError()` – kód uvnitř této metody se vykonává, jestliže dojde k chybě při sestavení textového řetězce
- `onTimeout()` – kód uvnitř této metody se vykonává, jestliže v definovaném čase neproběhla žádná promluva

Hlavní část reagování na promluvy se vykonává v metodě `onPartialResult()`, a to z důvodu nutnosti řešit příkazy v reálném čase. To znamená, že příkazy se vykonávají již v době promluvy – nečeká se na výsledek promluvy (metoda `onResult()`), kde je určitý nežádaný timeout, což by zapříčinilo významnou odezvu mezi promluvou a vykonáním příkazu.

Při implementaci rozpoznávače je v prvé řadě nutné pojmenovat si používané rozpoznávače. V této aplikaci jsou využívány dva rozpoznávače, pro rozpoznávání klíčových slov a frází – `COMMAND_SEARCH` a pro klíčové slovo k aktivaci naslouchání hlasových příkazů pomocí slova `listen` – `WAIT_FOR_KEYPHRASE`.

Při spuštění aktivity se vykonává metoda `onCreate()`, ve které je pro rozpoznávač nutné zavolat metodu `runRecognizerSetup()`, která inicializuje rozpoznávač. Inicializace je časově náročná operace, proto je vhodné ji vykonat pomocí `AsyncTask` – abstraktní třídy, která zaručuje vykonávání kódu na pozadí a postupně zobrazuje stav v UI vlákně. Na pozadí uvnitř metody `runRecognizerSetup()` se následně volá metoda `setupRecognizer()`, ve které se definuje akustický model rozpoznávače, používaný slovník a také se do rozpoznávače vkládá definovaný seznam klíčových slov a frází. Ve výpisu č. 5.1 je vyobrazen kód, jenž je obsažen v metodě `setupRecognizer()`.

```
1  private void setupRecognizer(File assetsDir) throws IOException {
2     recognizer = SpeechRecognizerSetup.defaultSetup()
3           .setAcousticModel(new File(assetsDir, "en-us-ptm"))
4           .setDictionary(new File(assetsDir, "cmudict-en-us.dict"))
5           .setKeywordThreshold(1e-5f)
6           .getRecognizer();
7      recognizer.addListener(this);
8
9      recognizer.addKeyphraseSearch(WAIT_FOR_KEYPHRASE, KEYPHRASE);
10
11     File commandList = new File(assetsDir, "keywords.list");
12     recognizer.addKeywordSearch(COMMAND_LIST, commandList);
13 }
```

Výpis 5.1: Implementace metody setupRecognizer()

Ve výpisu č. 5.1 je možné všimnout si, na řádku 5, nastavení prahu významnosti pro klíčové slovo `listen`. V uvedeném případě hodnotou `1e-5`, (písmeno `f`, značí datový typ `float`). Přidání klíčového slova `listen` do rozpoznávače `WAIT_FOR_KEYPHRASE` probíhá na řádku 9. Načtení souboru `keywords.list` s obsahem klíčových slov a frází probíhá na řádku 11. Přiřazení tohoto souboru k danému rozpoznávači `COMMAND_SEARCH` je potom vykonáno na řádku 12.

Reakce na rozpoznané příkazy jsou implementovány v metodě `onPartialResult()`, která je blíže popsána v kapitole 5.4.

5.4 Přiřazení hlasových příkazů k funkcím

Jediným parametrem metody `onPartialResult()` je instance třídy `Hypothesis`; zde pojmenovaná `hypothesis`. Úryvek těla metody `onPartialResult()` je vypsán ve výpisu č. 5.2.

```
1  public void onPartialResult(Hypothesis hypothesis) {
2        if (hypothesis == null)
3           return;
4        String text = hypothesis.getHypstr();
5        if (text.equals(KEYPHRASE)) {
6           isDone = false;
7           switchSearch(COMMAND_LIST);
8        } else {
9           switch (text) {
10             case "make a left move":
11                switchSearch(COMMAND_LIST);
12                if (!isMoveDone) {
13                   new Thread(new Runnable() {
14                      public void run() {
15                         moveCamera("left", tilt, pan);
16                      }
17                   }).start();
18                }
19                isMoveDone = true;
20                break;
21          }
22       }
23    }
```

Výpis 5.2: Úryvek těla metody onPartialResult()

Úryvek těla metody `onPartialResult()` obsahuje pouze kontrolu nad příkazem `make a left move` zajišťujícím pohyb kamery doleva. Získání textového řetězce z rozpoznané promluvy je vyobrazeno na řádku 4, ve výpisu č. 5.2. Na řádku 7 si lze všimnout volání metody `switchSearch()` s parametrem `COMMAND_SEARCH`, čímž změníme konfiguraci rozpoznávače z poslouchání klíčového slova `listen` na poslouchání klíčových slov a frází ze souboru `keywords.list`.

V případě, že se textový řetězec `text` rovná klíčové frázi `make a left move`, vykoná se zavolání metody `moveCamera()`, která je popsána ve výpisu č. 5.3.

```
1  public void moveCamera(String direction, float tilt, float pan){
2        switch (direction){
3           case "left":
4              while(!isDone){
5                 try {
6                    mSkyControllerDrone.cameraMove(tilt, pan);
7                    pan -= pref.getFloat("cameraspeed", 0.05f);
8                    Thread.sleep(5);
9                    setCorrectCoordinates(tilt, pan);
10                } catch (InterruptedException e) {
11                   e.printStackTrace();
12                }
13             }
14             break;
15       }
16 }
```

Výpis 5.3: Úryvek těla metody moveCamera()

Z výpisu č. 5.3 je patrné, že při zavolání metody `moveCamera()` s parametrem `direction` rovnajícím se řetězci `left`, se provede změna pozice kamery a to pomocí souřadnic `tilt` (vertikální rotace) a `pan` (horizontální rotace). Velikost změny v proměnné `pan` je definována podle uložených hodnot z nastavení v `SharedPreferences`, zajišťujícího uchování hodnot i po vypnutí aplikace (řádek 7).

Zavoláním metody `mSkyControllerDrone.cameraMove(tilt, pan)` s danými souřadnicemi se provede změna pomocí metody `setCameraOrientationV2()` implementované v `ARSDK`, volané dle konvence v podkapitole 4.2.1.3.

5.5 Video stream z dronu

Pro zpětnou vazbu pilota k pozici kamery je nutné získat video stream z dronu. Tento video stream by měl mít pilot neustále k dispozici, je vhodné jej tedy zakomponovat do layoutu aplikace v podobě pozadí. Řešením je vytvoření vlastního view, tedy komponenty, která má za úkol video stream zobrazovat. Ve výpisu č. 5.4 je vypsán úryvek z XML souboru `activity_skycontroller.xml`, zachycující vykreslení videostreamu v layoutu aplikace.

```
1       <cz.beles.andron.view.BebopVideoView
2          android:id="@+id/videoView"
3          android:layout_width="match_parent"
4          android:layout_height="match_parent"/>
```

Výpis 5.4: Úryvek layoutu aktivity SkyControllerActivity

Hodnoty `match_parent` v atributech `layout_width` a `layout_height` zajišťují roztáhnutí prvku tak, aby vyplňoval svého předka, v tomto případě `RelativeLayout`, který má rozměry vyplňující celou obrazovku.

Ve výpisu č. 5.4 vidíme prvek s identifikátorem `videoView`, který má jako svůj zdroj nastavenou třídu `BebopVideoView` z balíčku `view`. Tato třída je potomkem třídy `SurfaceView` zajišťující vykreslování vlastních grafických prvků (v tomto případě video streamu) v layoutu aplikace.

Pro video stream je použit video kodek standardu H.264. Rozlišení videa činí 640 x 368 px – nižší rozlišení videa činí aplikaci výkonnější. Nahrávané video po použití hlasového příkazu `record a video` je ovšem stále v plné Full HD kvalitě. Pro získání video streamu je využito třídy `ARFrame` z `ARSDK`, konkrétně metody `onFrameReceived()`.

5.6 Testování a optimalizace

Cílem testování a optimalizace je zjistit chybovost navrženého řešení aplikace a snížení či eliminace těchto chyb. Testování byla podrobena aplikace z pohledu rozeznávání a vykonávání hlasových příkazů. Je nezbytné si uvědomit, že aplikace využívá mikrofon mobilního zařízení a proto je vhodné podrobit testování na více zařízeních, jelikož citlivost mikrofonu se může, zařízení od zařízení, výrazně lišit.

Pro účely testování jsem využil dvě mobilní zařízení v tabulce č. 5.1.

Název zařízení	**Operační systém**	**CPU**
Asus Nexus 7 (2012)	Android 5.1.1	Cortex-A9 1,2 GHz
Huawei P8 Lite	Android 5.0.1	Cortex-A53 1,2 GHz

Tabulka 5.1: Zařízení, na kterých proběhlo testování aplikace [23, 24]

Jelikož jsou výsledky rozpoznávání náchylné na vnější vlivy, jako je vítr nebo hluk, podrobil jsem aplikaci testování ve dvou odlišných prostorách:

- V místnosti, uvnitř budovy

- Mimo obytné prostory, za průměrných povětrnostních podmínek[2]

Při každé z výše uvedených podmínek jsem testoval dvakrát stejnou sekvenci příkazů, u kterých jsem si následně poznačil, zda se příkaz úspěšně vykonal či nikoliv. Příkaz pro určení směru otáčení + příkaz `stop` v tomto případě považuji za jeden příkaz. Prováděné příkazy v pevném pořadí jsou následující:

1. Listen
2. Make a left move ... Stop
3. Make a right move ... Stop
4. Make an up move ... Stop
5. Make a down move ... Stop
6. Center
7. Take a picture
8. Record a video
9. End recording
10. Cancel

Seznam 5.1: Posloupnost seznamů použitých při testování aplikace

5.6.1 Výsledky testování

Výsledky testování jsou vyobrazeny v kontingenční tabulce č. 5.2. Jako typ testu jsem si pro větší přehlednost vytvořil zkratky v podobě písmen A ... D. Každé písmeno je dáno jako jedno unikátní testovací sestavení. Písmena A ... D značí:

- A ... Zařízení Nexus 7, vnitřní prostora
- B ... Zařízení Nexus 7, vnější prostor
- C ... Zařízení P8 Lite, vnitřní prostora
- D ... Zařízení P8 Lite, vnější prostor

Typ testu \ Rozeznání	**Ano**	**Ne**	**Celkem**
A	18	2	20
B	16	4	20
C	19	1	20
D	17	3	20
Celkem	70	10	80

Tabulka 5.2: Výsledky testování

[2] Za průměrné povětrnostní podmínky považuji slunečné počasí s větrem do 4 m/s.

Z tabulky č. 5.2 lze vyvodit pravděpodobnost úspěchu rozpoznání a vykonání příkazu u jednotlivých testovacích sestavení. Výpočty pravděpodobnosti úspěchu rozpoznání příkazu u jednotlivých testovacích sestavení jsou vyobrazeny na rovnicích č. 5.1 až 5.4.

$$P(A) = \frac{18}{20} = 0,90 \tag{5.1}$$

$$P(B) = \frac{16}{20} = 0,80 \tag{5.2}$$

$$P(C) = \frac{19}{20} = 0,95 \tag{5.3}$$

$$P(D) = \frac{17}{20} = 0,85 \tag{5.4}$$

V tabulce č. 5.3 jsou podrobněji zaznamenány úspěchy rozpoznání ke každému příkazu. Na ose X jsou zaznamenány testovací sestavení A ... D. Na ose Y jsou vypsány příkazy ze seznamu č. 5.1. Je nutné si uvědomit, že hodnoty v tabulce mohou nabývat hodnot z intervalu <0;2>, vzhledem k tomu, že se každý příkaz testoval u každého sestavení dvakrát.

		Testovací sestavení				**Celkem úspěchů**	**Celkem neúspěchů**
		A	**B**	**C**	**D**		
Číslo příkazu	**1**	2	2	2	2	8	0
	2	2	1	1	1	5	3
	3	2	2	2	2	8	0
	4	1	1	2	1	5	3
	5	2	1	2	2	7	1
	6	2	2	2	2	8	0
	7	2	2	2	2	8	0
	8	1	2	2	1	6	2
	9	2	1	2	2	7	1
	10	2	2	2	2	8	0

Tabulka 5.3: Počet úspěchů pro jednotlivé příkazy u daných tetovacích sestavení

Z výsledků z tabulky č. 5.2 lze vyvodit závěr, že nejspolehlivější testovací sestavení je sestavení s označením C, tedy mobilní zařízení Huawei P8 Lite při použití ve vnitřních prostorách, s pravděpodobností úspěchu rozpoznání a vykonání příkazu 95%. Naopak nejhůře v testu dopadlo sestavení s označením B, tedy Asus Nexus 7, při použití ve vnějším prostoru, s pravděpodobností úspěchu rozpoznání a vykonání příkazu 80%.

Vnější podmínky (mimo obytné prostory) jsou pro rozpoznávání řídících příkazů nepříznivé a je zde zcela na místě využití externích mikrofonů, pro lepší redukci vnějšího šumu.

Díky výsledkům, uvedeným v tabulce č. 5.3, je možné a vhodné optimalizovat rozpoznávání příkazů v některých konkrétních případech. U každého příkazu zaznamenaného v tabulce č. 4.4 je nezbytné experimentovat s danými prahy významnosti tak, aby byl výsledek co možná nejpřesnější. Dle tabulky č. 5.3 bylo nejméně příznivě vyhodnoceno rozpoznávání příkazů `Make a left move` a `Make an up move`. V obou případech se třemi chybami z osmi pokusů. Je tedy vhodné, v zájmu optimalizace kvality aplikace, u těchto příkazů pozměnit práh významnosti na vhodnější hodnotu.

Jelikož promluva nebyla v těchto případech rozpoznána, je nezbytné práh významnosti snížit (například z hodnoty `1e-10` na `1e-15`), tedy nastavit jej tak, aby byl rozpoznávač při kalkulaci v těchto dvou případech benevolentnější.

Pro použití aplikace v praxi je vhodné využít například bluetooth mikrofon, zavěšený za uchem, blízko úst. Tím by se značně snížila náchylnost na okolní ruch a hlasový vstup by byl mnohem přesnější a čistší. Dostupný bluetooth mikrofón pro tuto práci ovšem technologicky nevyhovoval standardům používaných aplikací, nebylo tedy možné tento mikrofon využívat k hlasovému zadávání příkazů.

Další možností, jak zvýšit kvalitu hlasového zadávání a redukovat okolní ruch, je použití hrdelního mikrofonu. Takový mikrofon nepodléhá povětrnostním podmínkám, jako je například šum větru, a jiným rušivým vlivům, například hluku vnějšího prostředí. Pro tento typ mikrofonu by bylo potřebné opětovně natrénovat akustický model tak, aby byl schopen rozpoznat dané příkazy, a to z důvodu jiného zpracování akustického signálu, než u klasických mikrofonů.

Závěr

Výsledkem této diplomové práce je realizovaná aplikace pro mobilní zařízení s operačním systémem Android, která umožňuje ovládání příslušenství dronu Parrot Bebop 2 pomocí hlasových příkazů.

Práce byla rozdělena na část teoretickou a praktickou. Teoretická část byla věnována historií UAV zařízení, principům funkce multikoptér, základům lidského hlasu, teorií systémů pro automatické rozpoznávání řeči, včetně jejich architektury, a následně teorií skrytých Markovových modelů a umělých neuronových sítí.

Praktická část byla věnována popisu komunikačního řetězce mezi vysílačem a dronem, za pomoci softwarových nástrojů ARSDK. Bylo zde popsáno využití nástroje CMU Sphinx, kde je zahrnut i popis vytváření jazykového modelu a fonetického slovníku. Dále byl v této části zpracován návrh a implementace aplikace, byl realizován návrh grafického prostředí, implementován rozpoznávač hlasu a k řídícím funkcím dronu byly přiřazeny jednotlivé hlasové příkazy. V závěru bylo provedeno testování a optimalizace aplikace.

V průběhu realizace diplomové práce projevila firma SkySystems Europe, s.r.o., zájem o oblast hlasového ovládání příslušenství dronů a v rámci inovačního voucheru, registrační číslo CZ.01.1.02/0.0/0.0/16_045/0007486, jsem se osobně podílel na projektu *Hlasové ovládání příslušenství dronu*, ve kterém jsem využil nabyté zkušenosti a vědomosti.

Mobilní aplikaci, jenž je předmětem této diplomové práce, lze nadále rozšiřovat o nové funkce (například o hlasové ovládání fyzického pohybu dronu). Aplikaci lze také dále optimalizovat pro použití hrdelního mikrofonu, čímž by se výrazně redukoval vnější šum způsobený povětrnostními podmínkami a dalšími nepříznivými vlivy vnějšího prostředí. Další zajímavou funkcí by mohla být kompatibilita aplikace s brýlemi pro virtuální realitu nebo natrénování aplikace pro hlasové příkazy zadávané v českém jazyce.

Při realizaci této diplomové práce jsem se setkal s velmi zajímavou oblastí rozpoznávání hlasu počítačem. Tato oblast má, dle mého názoru, velmi světlou budoucnost, a to nejen díky mobilním a nositelným zařízením, ale také díky automobilovému průmyslu či lékařství. Díky této diplomové práci jsem nabyl spoustu zkušeností a vědomostí pro praxi či k dalšímu studiu.

Bc. Miroslav Beleš

Literatura

[1] UHLÍŘ, Jan. *Technologie hlasových komunikací*. Vyd. 1. Praha: Nakladatelství ČVUT, 2007. ISBN 978-80-01-03888-8.

[2] YU, Dong a Li DENG. *Automatic speech recognition*. New York: Springer, 2014. ISBN 9781447157786.

[3] HASSOUN, Mohamad H. *Fundamentals of artificial neural networks*. Cambridge, Mass.: MIT Press, 1995. ISBN 02-620-8239-X.

[4] FREEDMAN, David. *Statistical models: theory and practice*. Cambridge: Cambridge University Press, 2005. ISBN 05-216-7105-1.

[5] RABINER, Lawrence R. a Biing-Hwang JUANG. *Fundamentals of speech recognition*. Englewood Cliffs: Prentice Hall PTR, c1993. Prentice Hall signal processing series. ISBN 01-301-5157-2.

[6] PHILIPS, Bill. *Android programming: the big nerd ranch guide*. 2nd edition. Atlanta, GA: Big nerd ranch, 2015. ISBN 978-0-13417-145-6.

[7] Přehled kanálů. *Kolmanl.info* [online]. 2008 [cit. 2016-12-19]. Dostupné z: *http://www.kolmanl.info/index.php?show=kmitocty_prehled*

[8] Jimmy Stamp. Unmanned Drones Have Been Around Since World War I. *Smithsonian.com*. [online]. 12. 2. 2003 [cit. 2016-01-30]. Dostupné z: *http://www.smithsonianmag.com/arts-culture/unmanned-drones-have-been-around-since-world-war-i-16055939/*

[9] Rise of the Multicopter. *Model Aviation.* [online]. 2012 [cit. 2016-03-11]. Dostupné z: *http://modelaviation.com/riseofmulticopter*

[10] RC Spread Spectrum Demystified. *RCHelicopterFun.com* [online]. 2014 [cit. 2016-12-21]. Dostupné z: *http://www.rchelicopterfun.com/RC-Spread-Spectrum.html*

[11] CLARK, James, Irena KOPRINSKA a Josiah POON. *A Neural Network Based Approach to Automated E-mail Classification* [online]. [cit. 2016-11-02]. Dostupné z: *http://testwww.it.usyd.edu.au/~josiah/wi03_email_classification.pdf*

[12] BRICH, Aleš. *Analýza robustnosti moderních rozpoznávačů řeči na bázi TANDEM architektury* [online]. 2016 [cit. 2016-11-04]. Dostupné z: *https://dspace.cvut.cz/bitstream/handle/10467/64773/F3-DP-2016-Brich-Ales-DP_Ales_Brich_2016.pdf*

[13] Jak se počítač učí rozpoznávat mluvenou řeč. *Osel.cz* [online]. 2010 [cit. 2016-11-04]. Dostupné z: *http://www.osel.cz/5152-jak-se-pocitac-uci-rozpoznavat-mluvenou-rec.html*

[14] *Korpus.cz* [online]. Praha: Ústav Českého národního korpusu, 1994 [cit. 2016-11-05]. Dostupné z: *https://www.korpus.cz/*

[15] VOLNÁ, Eva. *Neuronové sítě 1* [online]. Druhé vydání. Ostrava: Ostravská univerzita v Ostravě, 2008 [cit. 2016-11-05].Dostupné z: *http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf*

[16] PSUTKA, Josef. *Mluvíme s počítačem česky*. Praha: Academia, 2006. Česká matice technická (Academia). ISBN 80-200-1309-1.

[17] Parrot BEBOP 2. *Parrot* [online]. [cit. 2017-02-02]. Dostupné z: *https://www.parrot.com/ca/drones/parrot-bebop-2*

[18] PARROT SA. *ARSDK Protocols* [online]. 2015 [cit. 2017-02-03]. Dostupné z: *http://developer.parrot.com/docs/bebop/ARSDK_Protocols.pdf*

[19] Voice Controlled Drone with RasPi, Amazon Echo and 3DR IRIS+. *Hackster.io* [online]. 2016 [cit. 2017-02-07]. Dostupné z: *https://www.hackster.io/veggiebenz/voice-controlled-drone-with-raspi-amazon-echo-and-3dr-iris-c9fd2a*

[20] How to Build a Portable Voice Controlled Drone for Under $500. *Mobiquity.com* [online]. [cit. 2017-02-08]. Dostupné z: *https://www.mobiquityinc.com/innovation/how-to-build-a-portable-voice-controlled-drone-for-under-500*

[21] Perceptual Drone Speech Recognition. *Intel Developer Zone* [online]. 2015 [cit. 2017-02-09]. Dostupné z: *https://software.intel.com/en-us/articles/perceptual-drone-speech-recognition-using-realsense*

[22] Voice Controlled Arduino Drone. *Instructables* [online]. [cit. 2017-02-09]. Dostupné z: *http://www.instructables.com/id/SpeechVoice-Controlled-Arduino-Drone/*

[23] Huawei P8lite. *GSMArena* [online]. 2015 [cit. 2017-03-09]. Dostupné z: *http://www.gsmarena.com/huawei_p8lite-7201.php*

[24] Asus Google Nexus 7. *GSMArena* [online]. 2012 [cit. 2017-03-09]. Dostupné z: *http://www.gsmarena.com/asus_google_nexus_7-4850.php*

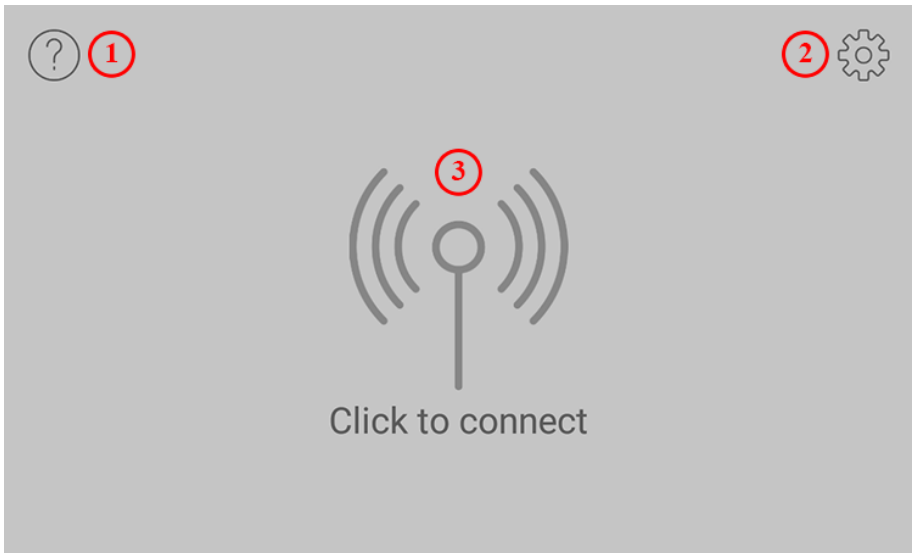
A Uživatelská dokumentace

Účelem této části je seznámit uživatele s funkcionalitou softwaru Andron a seznámit ho s ovládáním této aplikace. Funkce jednotlivých obrazovek jsou popsány v následující podkapitole č. A.1.

A.1 Popis jednotlivých obrazovek aplikace

Hlavní obrazovka

Účelem hlavní obrazovky je uživatelsky přívětivá navigace uvnitř aplikace. Obrazovka obsahuje tři tlačítka, jak je patrné z obrázku č. A.1. Tlačítko č. 1 A.1 je určeno pro přechod do sekce *Informace*, tlačítko č. 2 je určeno pro přechod do sekce *Nastavení*, tlačítko č. 3 je určeno k přechodu do sekce *Výběr zařízení*.



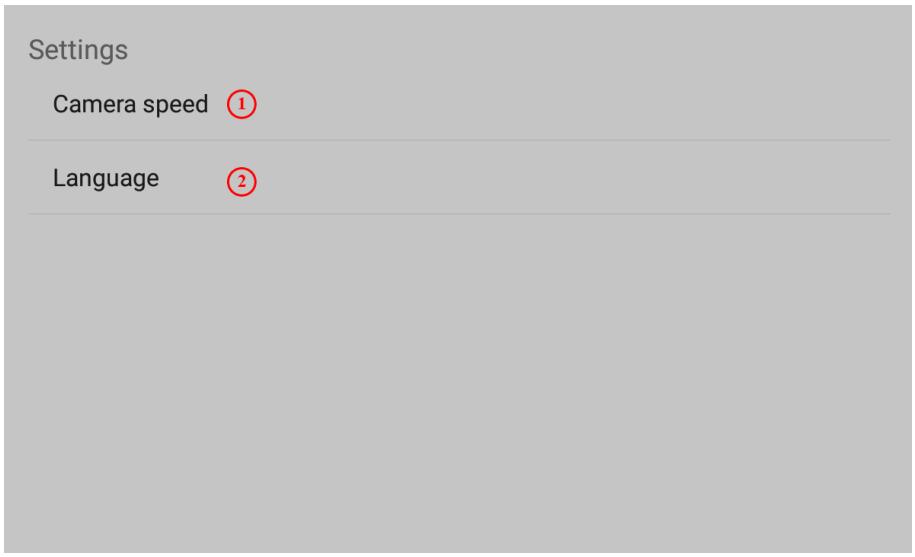
Obrázek A.1: Hlavní obrazovka

Nastavení aplikace

V nastavení aplikace má uživatel možnost nastavit:

- Rychlost otáčení kamery (bod č. 1 na obrázku č. A.2) – na výběr jsou tři možnosti:
  - 5°/s – Pomalé otáčení kamery
  - 10°/s – Středně rychlé otáčení kamery (výchozí)
  - 15°/s – Rychlé otáčení kamery
- Jazyk aplikace (bod č. 2 na obrázku č. A.2) – na výběr jsou k dispozici dva jazyky:
  - Anglický jazyk (výchozí)

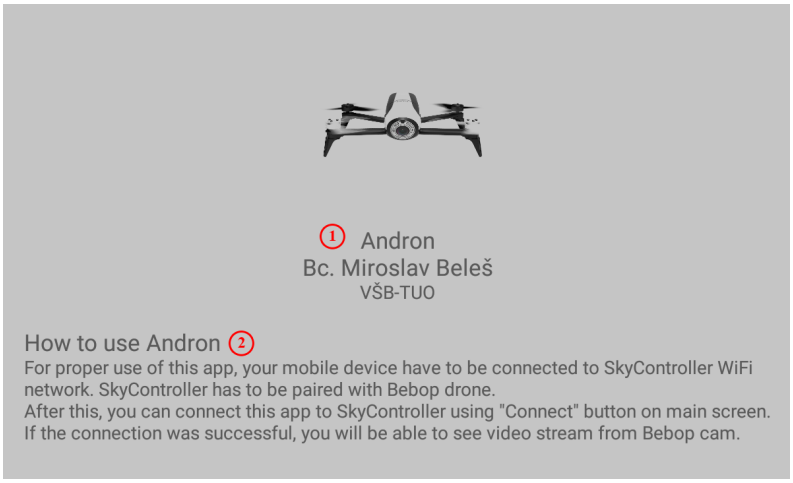
– Český jazyk



Obrázek A.2: Nastavení aplikace

Informační obrazovka

Funkcí obrazovky s informacemi o aplikaci je zobrazení informací o autorovi aplikace (bod č. 1 na obrázku č. A.3), dále obsahuje stručné instrukce k ovládání aplikace (bod č. 2 na obrázku č. A.3).

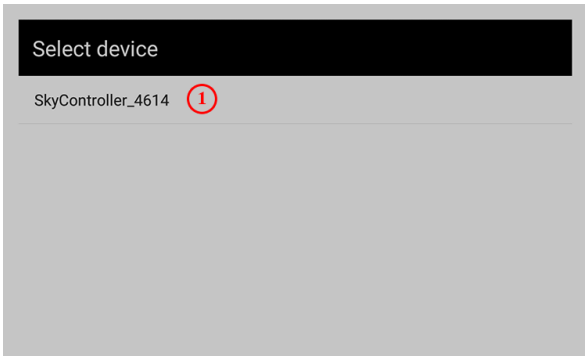


Obrázek A.3: Informační obrazovka

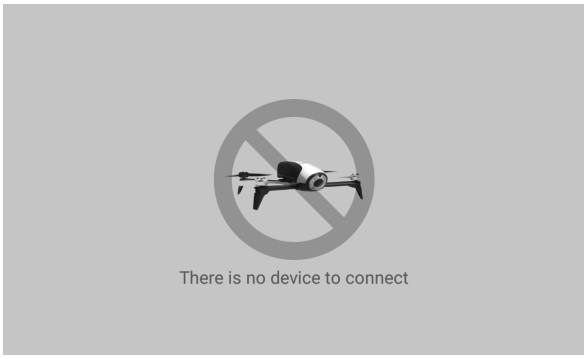
Obrazovka výběru zařízení

Po kliknutí na tlačítko č. 3, z obrázku č. A.1, se uživatel dostane na obrazovku výběru zařízení. Zde ze seznamu dostupných zařízení vybere zařízení k hlasovému ovládání. Bod č. 1 na obrázku č. A.4 zobrazuje příklad zařízení připraveného k párování se s aplikací.

V případě nenalezení zařízení v dosahu se zobrazí varovná hláška (obrázek č. A.5).



Obrázek A.4: Výběr zařízení



Obrázek A.5: Zařízení nenalezeno

Obrazovka „kokpit“

Po úspěšném připojení k zařízení se uživateli zobrazí video stream z dronu v reálném čase. V bodě č. 1, na obrázku č. A.6, se nachází stav baterie vysílače SkyController a dronu Bebop 2. Zda dron nahrává či nenahrává video, uvádí bod č. 2. Tato ikonka se může zobrazit ve dvou stavech:

- Bílá ikona – video se nenahrává
- Červená ikona – video se nahrává

Bod č. 3 značí stav rozpoznávače hlasu. Více informací o příkazech pro ovládání kamery se nachází v kapitole č. A.2.

Obrázek A.6: Obrazovka kokpit

A.2 Hlasové příkazy aplikace

Po úspěšném připojení k dronu může uživatel ovládat kameru dronu Bebop 2 hlasově. Pro aktivaci rozpoznávače hlasu je zapotřebí říct klíčové slovo *listen*. Po tomto klíčovém slovu aplikace naslouchá příkazům určeným k ovládání kamery. Možnosti manipulace s kamerou jsou následující:

- Ovládání otáčení kamery ve čtyřech směrech – nahoru, dolů, doprava, doleva
- Vycentrování kamery
- Vyfocení fotografie z kamery
- Spuštění a ukončení nahrávání videa z kamery

Tyto funkce se spouštějí hlasovými příkazy sepsanými v tabulce č. A.1.

Příkaz	**Popis**
make a left move	Otáčení kamery doleva
make a right move	Otáčení kamery doprava
make an up move	Otáčení kamery nahoru
make a down move	Otáčení kamery dolů
take a picture	Pořízení snímku z kamery
stop	Zastavení otáčení kamery
center	Vycentrování kamery
record a video	Započetí nahrávání videa
end recording	Zastavení nahrávání videa

Tabulka A.1: Seznam hlasových příkazů k ovládání kamery dronu

Pro deaktivaci naslouchání příkazů stačí pronést klíčové slovo *cancel*. Příklad takového zadávání příkazu může být následující:

1. Listen
2. Record a video
3. Make a right move
4. Stop
5. Take a picture
6. Make an up move
7. Stop
8. Take a picture
9. End recording
10. Cancel

Podle uvedeného příkladu se kamera dronu bude chovat následovně:

Po započetí rozpoznávání kamery slovem *listen* se spustí nahrávání videa (*record a video*), poté se kamera posune doprava (*make a right move*), pohyb se zastaví (*stop*), kamera pořídí fotografický snímek (*take a picture*), následuje posun směrem nahoru (*make an up move*), pohyb kamery se zastaví (*stop*), je pořízen další fotografický snímek (*take a picture*), nahrávání videa je ukončeno (*end recording*). Ukončením nahrávání videa je deaktivován rozpoznávač hlasu (*cancel*), je ukončeno naslouchání příkazům, určeným k manipulaci kamerou.

B Programová dokumentace

Veškeré zdrojové kódy a potřebné přílohy jsou k dispozici v přiloženém ZIP souboru `BEL0075_FEI_N2647_2612T059_2017_priloha.zip`. Adresářová struktura tohoto ZIP souboru je následující:

```
<ZIP>
├── Andron
│   ├── .grandle
│   │   └── 2.14.1
│   ├── .idea
│   │   ├── copyright
│   │   └── libraries
│   ├── build
│   │   ├── generated
│   │   └── intermediates
│   └── grandle
│       └── wrapper
├── app
│   ├── build
│   │   ├── generated
│   │   ├── intermediates
│   │   ├── outputs
│   │   └── tmp
│   ├── libs
│   └── src
│       └── main
└── Doc
    ├── cz
    └── index-files
```

V adresáři `Doc` se nachází programová dokumentace vygenerovaná pomocí nástroje Javadoc. Vzhledem k tomu, že byl projekt implementován ve vývojářském prostředí Android Studio 2.2.3, je vhodné kompilovat projekt v prostřední Android Studio ve verzi 2.2.3 nebo vyšší.

Programová dokumentace nástroje CMU Sphinx, konkrétně balíčku Pocketsphinx je k dispozici zde: `http://cmusphinx.sourceforge.net/doc/pocketsphinx/`

Programová dokumentace Parrot SDK je k dispozici zde: `http://developer.parrot.com/docs/SDK3/`